Crystalline: Lowering the Cost for Developers to Collect and Organize Information for Decision Making

Michael Xieyang Liu
Human-Computer Interaction Institute, Carnegie Mellon University
Pittsburgh, PA, USA
xieyangl@cs.cmu.edu

Aniket Kittur
Human-Computer Interaction Institute, Carnegie Mellon University
Pittsburgh, PA, USA
nkittur@cs.cmu.edu

Brad A. Myers
Human-Computer Interaction Institute, Carnegie Mellon University
Pittsburgh, PA, USA
bam@cs.cmu.edu

ABSTRACT

Developers perform online sensemaking on a daily basis, such as researching and choosing libraries and APIs. Prior research has introduced tools that help developers capture information from various sources and organize it into structures useful for subsequent decision-making. However, it remains a laborious process for developers to manually identify and clip content, maintaining its provenance and synthesizing it with other content. In this work, we introduce a new system called Crystalline that automatically collects and organizes information into tabular structures as the user searches and browses the web. It leverages natural language processing to automatically group similar criteria together to reduce clutter, and uses passive behavioral signals such as mouse movement and dwell time to infer what information to collect and how to visualize and prioritize it. Our user study suggests that developers are able to create comparison tables about 20% faster with a 60% reduction in operational cost without sacrificing the quality of the tables.

CCS CONCEPTS

• **Information systems** → **Decision support systems**; • **Software and its engineering** → *Software design tradeoffs*; • **Human-centered computing** → Graphical user interfaces.

KEYWORDS

Sensemaking, Developer tools, Decision making, Behavior patterns, Implicit signals

ACM Reference Format:
Michael Xieyang Liu, Aniket Kittur, and Brad A. Myers. 2022. Crystalline: Lowering the Cost for Developers to Collect and Organize Information for Decision Making. In *CHI Conference on Human Factors in Computing Systems (CHI '22), April 29-May 5, 2022, New Orleans, LA, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3491102.3501968

1 INTRODUCTION

Developers spend a large portion of their time searching and making sense of the web for solutions to their programming problems [9, 108]. In many cases, the answers to such problems are not limited to a single solution, but developers discover that there are multiple legitimate options, and they must identify relevant criteria and constraints based on their unique contexts and carefully consider the trade-offs among those possible options [42, 63, 77, 78, 81, 82, 92, 94, 100, 107]. For example, when converting an old web application to use a modern JavaScript front-end framework, React.js [34] (with its ability to be progressively adopted into existing code bases) may be more suitable when one wants to gradually convert each separate module while minimizing the overall system downtime, whereas a more comprehensive framework such as Angular [47] might be a better choice if one wants to take advantage of various official utility packages like routing [44], animation [45] and data validation [46].

There have been many commercial and research tools and systems that try to help people make sense of information about tradeoffs to facilitate further decision making, such as by helping with easily capturing snippets of information [1, 5, 53, 110, 121] from web pages or organizing and synthesizing information into useful schema and representations [15, 29, 61, 71, 81, 122]. For example, one common practice that people employ is copying pieces of text as well as taking screenshots and putting them in a running Google Doc as they search and browse the web [88]. One system that is relevant to the context of programming is Unakite [81], which enables developers to collect and organize information online into comparison tables with options, criteria, and evidence to help with making decisions (see Figure 2).

However, even with the above tools, it remains a challenging process for developers to *manually* identify and capture the relevant content, maintain its provenance (where it came from), and synthesize it with other content. Prior work suggests that one cause is that people are often uncertain about which information will eventually turn out to be relevant, valuable, and worth capturing, especially at early stages of their learning and exploration when they are overloaded with information [4, 37]. Under these circumstances, people are hesitant to frequently pause and shift their focus from the investigation itself to reasoning about what to capture for later use [14, 58, 72, 109], or they could be too engaged in the sensemaking process and forget to collect anything at all. Indeed, research suggests that interactions for gathering information while performing active reading need to be quick and low effort, otherwise people tend not to capture information in the first place [58, 81, 85, 118]. In addition, though existing tools provide users with the flexibility and agency to synthesize the collected information into useful representations, such as comparison tables [15, 81] or knowledge maps [87], developers still need to perform these organizing operations manually. This is often a laborious process, as developers need to

CHI '22, April 29-May 5, 2022, New Orleans, LA, USA

ACM ISBN 978-1-4503-9157-3/22/04.
https://doi.org/10.1145/3491102.3501968

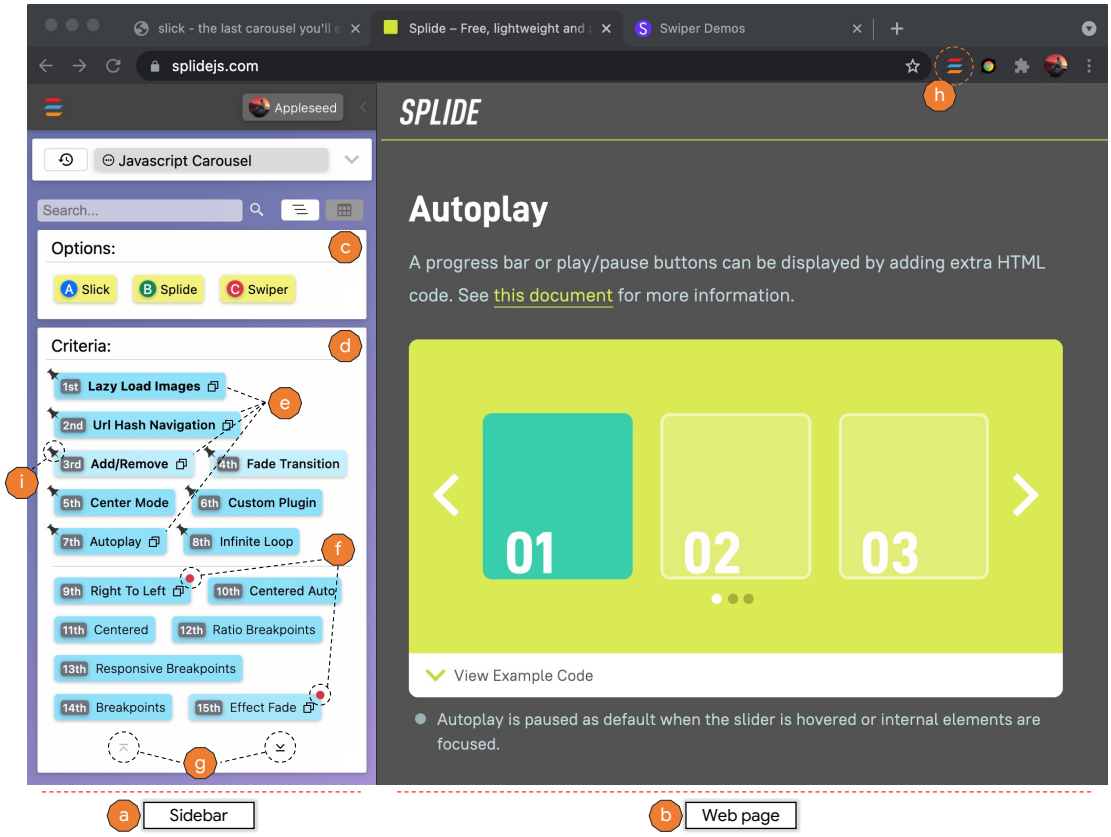

Figure 1: Crystalline's list view UI (a). As the developer browses a web page (b), Crystalline attempts to automatically collect options and criteria from the page, and display them in the options (c) and criteria panes (d) in the sidebar (a). In addition, Crystalline leverages natural language processing to automatically group similar criteria together, as shown by the multiple-pages icon (e). Crystalline uses behavioral signals such as mouse movement and dwell time to try to automatically detect the relative importance of the criteria (shown by the display order, with most important at the top). Users can use the "See more" and "See less" buttons (g) to adjust how many criteria are to be displayed at once. Crystalline will remind users of the existence of additional related evidence through a red notification dot at the top right of a criterion (f). The sidebar can be toggled in and out by clicking the browser extension icon (h). Users may pin (i) important criteria to the top of the list.

take stock of all the pieces of information, identify connections among them, and directly manipulate the representation to reflect the connections.

Another challenge reported in prior work is that developers' needs for collecting and organizing information are often not discovered until part of the way through an investigation process [16, 81]. This could be due to several major reasons, including but not limited to: 1) additional external requirements, constraints, or user feedback are discovered or introduced in the middle of a project which significantly complicates the original decision making problem [23, 30, 31]; 2) developers discover many more options, criteria, and their trade-offs than they anticipated at the beginning [81]; and/or 3) developers are required to explain or document their decisions and design rationale after the fact for the long-term maintainability and success of a software project [25, 39, 75, 76, 79, 104, 112]. In these situations, it is hard and involves duplicate work for developers to recall and retrace their steps for reaching their current state of sensemaking (the linear history visualization in almost all current browsers is known to be not particularly effective [16, 67, 124]) and recollect all the relevant evidence again.

In our new work, we explore the idea of having a system dynamically help users keep track of and organize information by leveraging the content they are browsing and the signals from their browsing behavior. Although we focus on the domain of programming due to strongly motivating prior work and ease of prototype development due to regularities of the programming context, our work may also generalize to other sensemaking contexts on the web. We instantiate this idea in a prototype system called Crystalline,[1] which is an extension to the Chrome web browser. Crystalline plays the role of a user's copilot and attempts to automatically identify and keep track of the options, criteria, and the corresponding evidence snippets from the web pages that a user has viewed, and organize the snippets into both list and tabular formats. To achieve

[1] Crystalline is named after rocks made up of interlocking crystals. It stands for **C**lipping **R**esulting in **Y**our **S**tructure as **T**ables **A**nd **L**ists **L**inked to **I**mplicit **N**otetaking **E**asily.

this, Crystalline mines a variety of behavioral signals while a user browses the web, including scrolling patterns and mouse cursor actions, and employs natural language understanding techniques to automatically classify and organize the collected content. The goal is that users can focus more on reading and understanding web content while occasionally guiding the system when it makes mistakes. We conducted a user study to evaluate the usability and effectiveness of Crystalline compared to Unakite as a baseline, which found that developers are able to build comparison tables about 20% faster with a 60% reduction in operational cost without sacrificing the quality of the tables. In particular, it only requires around 12% of the total task completion time for participants to use the tool to build and maintain a table, compared to around 30% in the baseline condition.

The primary contributions described in this paper include:

- evidence that it is possible to automatically identify options, criteria, and relevant evidence from web pages that a user is browsing using a set of natural language understanding heuristics,
- a set of implicit behavioral signals that users exhibit when browsing the web which can be used for prioritizing and filtering that collected information,
- a prototype system called Crystalline that integrates the heuristics and signals to automatically collect and organize viewed information into list and comparison table views for subsequent decision making,
- an evaluation that offers empirical insights into the usability, usefulness, and effectiveness of those signals and the system.

2 RELATED WORK

2.1 Sensemaking in Software Development

Sensemaking is widely considered to be the process of searching, collecting, and organizing information to iteratively develop a mental model that best fits the evidence [96, 106]. As knowledge workers [9], many activities that developers perform on a daily basis involve extensive sensemaking, such as designing the overall software architecture [56, 83], learning and understanding unfamiliar code and concepts [26, 73], debugging and fixing incorrect software behaviors [25, 74], planning and executing code refactorings [32, 41, 86], and evaluating past code and design patterns for future reuse [82, 91]. In this work, we focus on the particular type of sensemaking activity where developers leverage web resources to make a *decision* to solve their programming problem [9, 63]. Here, developers not only need to find information pertinent to their problem [8, 59, 97, 115], which is the first step in such complex sensemaking tasks [106, 123], but also collect and synthesize relevant information into structured knowledge so that they can make progress towards fully understanding the decision space [53, 71, 72, 81]. Indeed, our survey [63] revealed that over half of the questions asked on Stack Overflow contain answers with multiple options, each option valuable to the programming community due to a unique set of criteria that it fulfills.

Software engineering research has also identified that subsequent developers frequently need help with understanding the rationale of design decisions and code implementations made by previous developers [75, 76, 112]. This can be particularly difficult if the previous developers failed to properly document the rationale [120], or the documentation was incomplete or not up-to-date [38]. Granted, the fundamental challenge here is that it is effort- and time-intensive for decision authors to document their rationale (either in situ or after the fact) with little immediate payoff for themselves [42]. Our previous Unakite tool [81] addressed this challenge by encouraging authors to document their decision making processes and results using the tool's lightweight collecting and organizing features. Building on top of this, Crystalline further transforms the previously active capturing and organizing work [5, 81, 110] into passive monitoring and error-fixing [80], which has been shown to present a much lower entry barrier for people to start contributing [37].

2.2 Tools for Collecting and Organizing Information

To help people more effectively gather and process online information, systems and tools like SenseMaker [3], SearchPad [5], Hunter Gather [110], CoSense [93], Tabs.do [16], as well as commercial systems like the Evernote clipper [33], enable people to take entire pages or snippets of content from the web, classify them, and later put them together into a document with a coherent narrative for sensemaking, decision making or sharing and collaboration. However, one common characteristic of these tools is that it is mostly the user's responsibility to *manually* complete the information collection, triage, and organization process, while we attempt to do this *automatically* with Crystalline as the user searches and browses the web.

Other threads of prior research have explored different ways for machines to help during sensemaking, which inspired and informed our design. For example, systems like Entity Quick Click [6, 66, 116] employ techniques like named-entity recognition [84] to pre-process and highlight semantically meaningful entities in web content, and enable users to collect and annotate relevant information with a single click. Previous work like Thresher [60] and Dontcheva et al.'s personal web summarization tool [29] let users annotate and curate patterns and templates of information that they would like to collect on a few example web pages, then automatically collect them from future pages. In addition, Chang et al.' Mesh system [15] automatically retrieves relevant consumer product facts and reviews from Amazon into a comparison table to enable users to curate and explore nuanced options and criteria. These systems have largely relied on natural language understanding to analyze and transform the web content that users browse and read, while we argue that leveraging the signals from users' natural browsing behavior, such as dwell time and cursor movements, would unlock a new design space for automated machine support during online sensemaking, motivating us to use both NLP heuristics and passive behavioral signals to infer what information to collect and how to visualize and prioritize it in Crystalline.

2.3 Implicit Behavioral Signals When Using the Web

Prior research has investigated various implicit behavioral patterns that people exhibit when reading and interacting with content on a digital screen. One thread of research has explored using behaviors

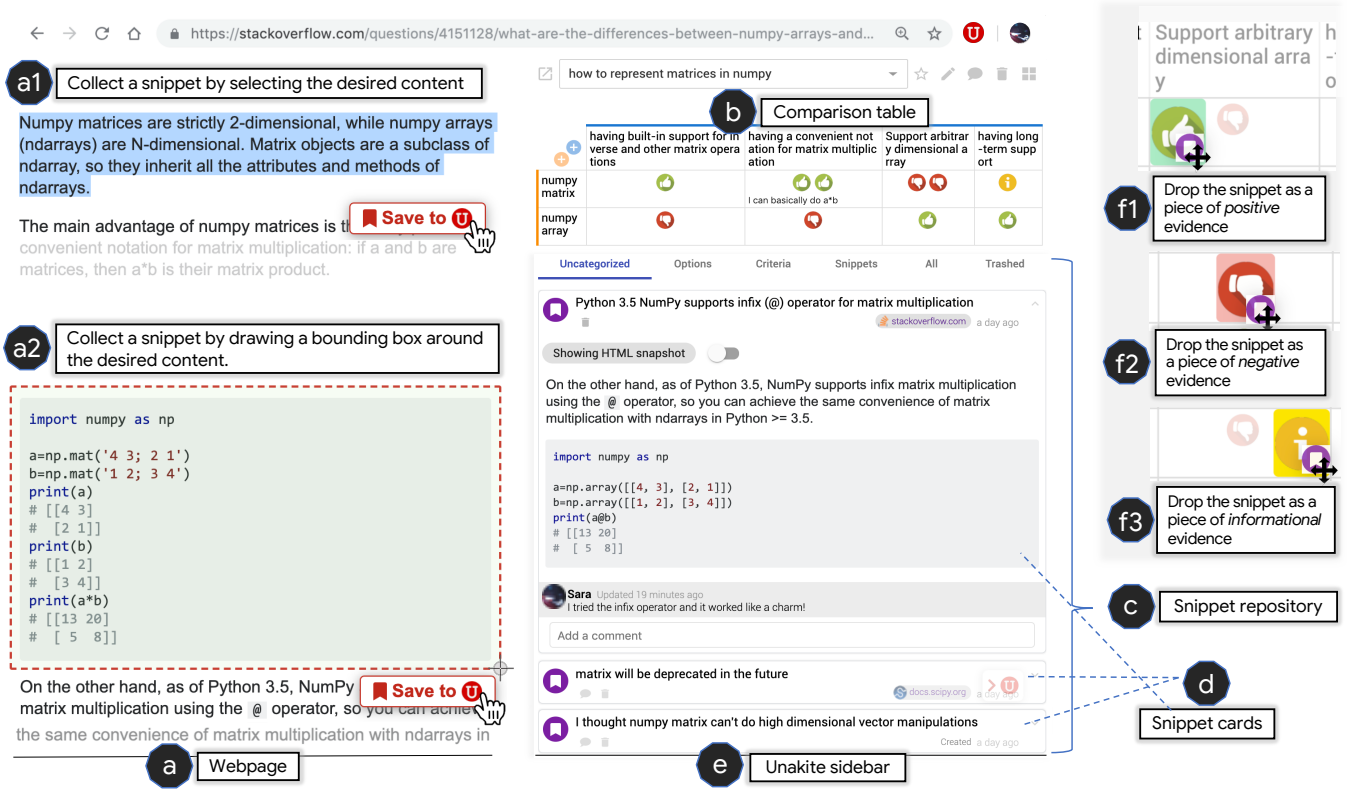

Figure 2: Unakite's user interfaces. With Unakite, a developer collects snippets by selecting the desired content (a1) or by dragging out a bounding box around the desired content (while holding the `Option` / `Alt` key) (a2) and clicking the "Save to U" button. The collected snippets will show up under the "Uncategorized" tab in the snippet repository (c) inside the Unakite sidebar (e). The developer can drag a snippet and drop it in one of the cells in the comparison table (b), and mark whether it is positive (green thumbs-up) or negative (red thumbs-down) or just informational (yellow "i"). (f1-f3) show the details of the three parts of each cell in the table where the snippet can be dropped. This figure is adapted from [81]. For full details, see [81].

such as dwell time, cursor movements, clicks, scrolling patterns, and gaze positions as *implicit signals* to approximate user interest on web pages as well as search result relevance [22, 50, 51, 57, 65]. For example, Claypool et al. [22] had participants use a custom-built browser to surf the web and concluded that the time spent on a page, the amount of scrolling on a page, and the combination of time and scrolling had a strong correlation with explicit user interest. In addition, Hijikata [57] discovered that actions such as text tracing and link pointing are decent behavioral indicators for perceived interesting segments of web pages. Similarly, in the domain of web searches, Buscher et al. [10–12], Guo and Agichtein [50, 51], and Huang et al. [65] demonstrated that eye tracking, as well as interactions like scrolling and cursor hovers, could accurately predict user interests in search results pages.

Building on the empirical understanding laid out by this research, in this work, we explore putting a combination of these implicit behavioral signals into use to approximate user visual attention in a working prototype. We used heuristics and pilot testing to devise mechanisms that translate the raw behavioral signals into numeric scores representing the "amount of attention" a user has given to a particular piece of online content. We then use these scores to filter out and rank the content of the evolving comparison table, further reducing the cost for developers to manually manage and prioritize collected information incrementally as they are searching and browsing.

3 BACKGROUND AND DESIGN GOALS

In this work, we explore automatically keeping track of and organizing relevant information on the web about trade-offs for developers as they are making decisions. To ground our research, we build on the "Option-Criterion-Evidence" framework introduced in our Unakite system [81]. We first briefly explain this framework as well as the Unakite system to provide necessary background for this research. Then we discuss the design goals for the new Crystalline system.

3.1 The Unakite System

Unakite was designed to address both the need of developers to synthesize online information about trade-offs when making

programming decisions as well as the need of subsequent developers to be able to understand the rationale behind those decisions [81]. As a Chrome extension, Unakite enables developers to manually collect any content from any web pages as *snippets* (pieces of information, Figure 2-d) into the snippet repository (a holding tank of information snippets, Figure 2-c) by selecting (Figure 2-a1) or dragging out a bounding box to enclose the desired content with the mouse cursor (Figure 2-a2). To organize the collected content, developers can use drag-and-drop to move the collected snippets from the repository into a comparison table (Figure 2-b) *options* (as row headers, e.g., a solution to solve a problem), *criteria* (as column headers, e.g., a standard by which options are judged), and *evidence* ("thumbs-up" or positive, "thumbs-down" or negative, and "informational" ("i") ratings that spread across the rest of the table cells) that illustrates the trade-offs among various options on those criteria. Developers can also rank the options and criteria in the table to reflect their unique order of preferences. The resulting comparison table is automatically saved and can be used by subsequent developers to understand the context of the previous decision space: what options and alternatives were explored, what criteria needed to be met, what trade-offs were discovered, and what was considered the most important and why.

Although Unakite has been shown to incur less operational overhead when it comes to collecting and organizing information in situ compared to common baseline methods like using Google Docs [81], developers still need to manually collect and structure each piece of content, which can be a costly process [58, 71, 72, 85, 118]. In addition, it forces developers to start using the tool from the outset to be able to capture the whole exploration, but, for cases in which the needs for collecting and organizing information are not discovered until partway through an investigation process (which can be quite common in agile style software development [23, 30, 31, 81] that is widely adopted across the software development industry), developers would have to retrace their exploration paths from the beginning and re-collect and organize the content, wasting time and causing duplicate work.

3.2 Design Goals

In order to address the above limitations of Unakite as well as other similar sensemaking tools [3, 5, 16, 93], we formulated the following design goals:

- **Minimize the cost to collect information.** The system should attempt to automatically collect information in the background without the user's specific attention or direction. This will help users focus on the main task of reading and comprehending the content.
- **Actively filter, organize, and prioritize information.** The system should actively filter, organize, and prioritize the collected information that gets presented to the user and help the user avoid information overload.
- **Reduce the cost of incorrect automation support.** In cases where machine support is incorrect or undesirable, the system should allow users to easily recover from those mistakes [2, 62].

4 CRYSTALLINE

4.1 System Overview

Guided by prior work and our design goals, we designed and implemented Crystalline, a Chrome extension prototype to help developers automatically collect and organize information relevant to their decision making problems.

Users mainly interact with Crystalline through a **sidebar** (Figure 1a) that is injected directly into every web page. As a developer opens and reads web pages, the sidebar will be updated with the automatically collected options (Figure 1c) and criteria (Figure 1d) in the *list view* (Figure 1c & d). The list view serves as a concise and glanceable outline that reflects one's exploration progress — what options one has encountered and what criteria one has looked into. Clicking on one of the criteria will enter a detailed view for that criterion (Figure 3a), listing out all the collected evidence snippets organized by options; similarly, clicking on an option will enter the *detailed view* for that option, which lists all the related criteria and the corresponding evidence associated with that option. Details on how we currently implemented the automatic collection and organization features are discussed in section 4.2.

In addition, developers can also switch to the *comparison table view* (Figure 3c) that summarizes the decision making space and the trade-offs among various options in detail. The order in which a criterion gets presented both in the list and the comparison table view are based on the *estimated importance* of the item to the user, which we approximate by the *amount of attention* a user has given to it. This, in turn, is derived from the user's implicit behavioral signals, which we will discuss in detail in section 4.2.2. To examine a particular piece of evidence in the detailed view or a comparison table cell, users can hover on it to zoom in (Figure 3b), or click on it to *teleport* to the original web page and scroll position from where it was previously collected.

Similar to previous systems [61, 81, 99], the sidebar can be toggled in and out like a drawer by clicking the extension icon (Figure 1h) or using a keyboard shortcut. Developers can passively monitor the sidebar as they are searching and browsing to make sure the system performs correctly, and quickly correct or dismiss the mistakes that the system makes. In addition, developers are free to hide the sidebar to have an unobstructed view of the web page, knowing that all the features for automatic information collection and organization are still running in the background even if the sidebar is in the hidden state.

4.2 Detailed Design

We now discuss how the different features in Crystalline are designed and implemented, and how they support our design goals.

4.2.1 Collecting information about options and criteria. In Crystalline, we explore having the system *automatically* collect relevant information in the background without the user having to explicitly perform the action of collecting information. This has the benefit of minimizing the distraction and cost of keeping track of information as an extra step in addition to thinking about the content on a web page, which, in turn, maximizes a user's attention to reading and understanding the content itself.

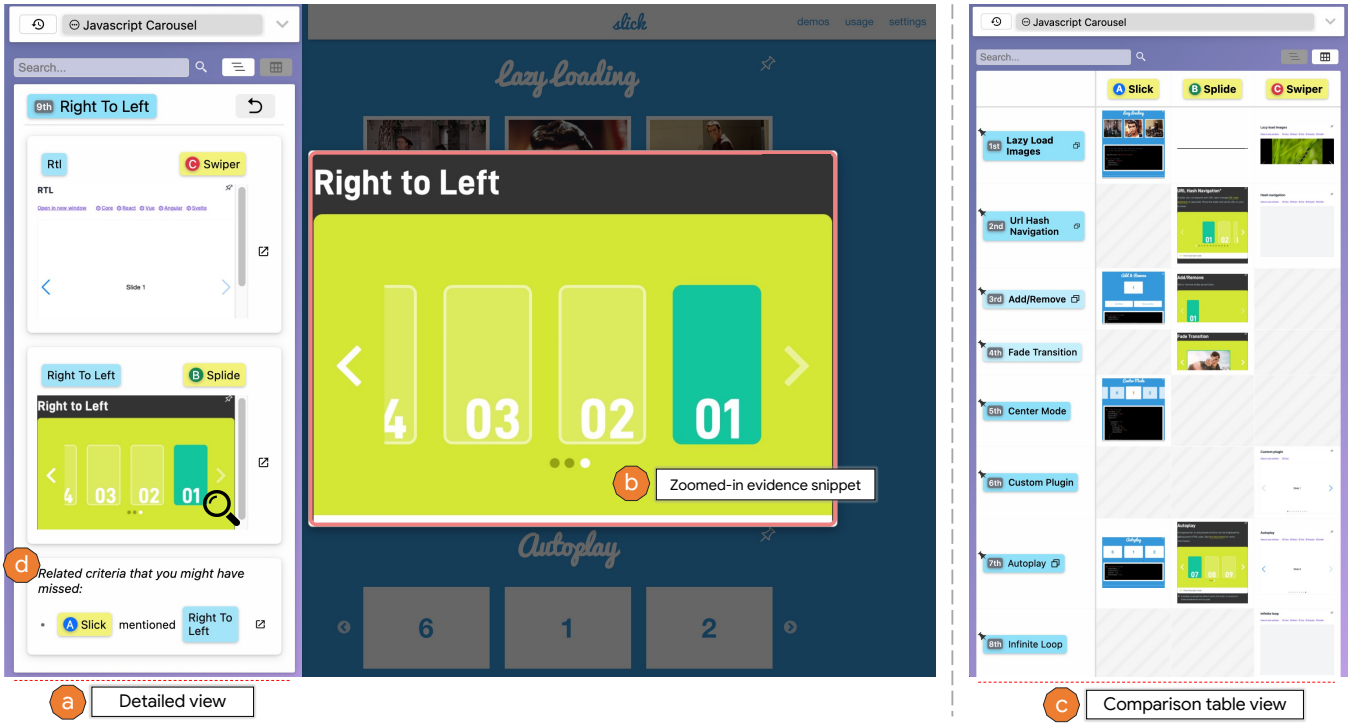

Figure 3: Additional Crystalline's user interfaces. Clicking on one of the criterion in the criteria pane (Figure 1d) will enter a detailed view for that criterion (a), listing out all the collected evidence snippets organized by options. Users can zoom in on an evidence snippet (b) by moving the mouse cursor over it in the detailed view until the cursor becomes a magnifying glass. Crystalline will actively look for and remind users of evidence for the same or similar criteria from pages that users have visited but have not yet paid attention to (d). Finally, similar to Unakite [81], Crystalline offers a comparison table view (c) that summarizes the decision making space and the trade-offs among various options in detail.

Specifically, Crystalline collects information about options, criteria, and their associated evidence snippets as discussed previously, which was reported by prior work as the key aspects developers look for when solving decision making problems [63, 75, 81]. Currently, to automatically recognize the *options*, Crystalline employs the following techniques: (1) it looks for the word or phrase between any instances of "vs." (or other variants like "v.s.", "versus", etc.) in web page titles and opening paragraphs and adds them as potential options. For example, the Medium.com article titled "Tensorflow vs Keras vs Pytorch: Which Framework is the Best?"[2] would yield "Tensorflow", "Keras", and "Pytorch" as three potential options; (2) it first runs noun phrase and entity extractions using the Google Cloud Natural Language API [48] on the web page title, section headers as well as the column and row headers of any HTML tables, then checks if the identified entities are mentioned in the titles of other visited pages. In addition, it also checks if the identified entities would frequently come up in each other's Google autocomplete results (the Google "vs" technique is described in [40, 82], which issues queries in the form of "`[option_name] vs`" to the Google Autocomplete API to get a list of autocomplete results that can be interpreted as potential alternatives to "`[option_name]`". An earlier version of this technique was launched as an experimental feature named Google Sets [21, 119]). Furthermore, it checks if the identified entities are mentioned repeatedly across the main content of the current web page. All potential options will go through a final deduplication process to produce the final list of options presented in the *options pane* (Figure 1c) in the sidebar. We chose and tuned these heuristics based on our internal usage and pilot testing results. In the future, more advanced NLP techniques could be used to augment the current set of heuristics.

Crystalline uses a similar set of heuristics to identify *criteria* from the web pages, with an emphasis on examining section headers and table headers (and entities extracted from them) rather than website titles. In this work and in the context of programming, we focus on using such heuristics to identify the criteria directly mentioned in the content, such as extracting "learning curve" from "React is widely considered to have quite a steep learning curve." We leave the extraction of latent criteria for future work, which are more commonly seen in domains other than programming, such as extracting "price" from "I bought this mp3 player for almost nothing" [98].

Further, users can always edit the options and criteria names, delete unwanted options or criteria, or manually select and collect any text as either an option or a criterion using the popup menu (Figure 4) as a backup.

[2]https://medium.com/@AtlasSystems/tensorflow-vs-keras-vs-pytorch-which-framework-is-the-best-f92f95e11502

Implicit Behavioral Signal	Selected References in Prior Research	Descriptions	Strength of indication of user attention	Score Function W
Copying content	Developers frequently copy sample code from the web to use in their own code [8, 54, 55]	Triggers when the user copies some text from a content block b. This typically happens when a developer copies sample code from web pages to try out in their own code.	Strongest	40 for each triggering
Text highlighting	People tend to highlight text while reading to help focus their attention [105]	Triggers each time when some text in a content block b gets selected. Triggerings where the selected text is shorter than 5 characters are disqualified.	Strong	20 for each triggering
Clicking	Clicking on content, such as widgets and links, is considered to be a decent behavioral indicator for perceived interesting elements on web pages [57]	Triggers when the user clicks on a content block b. This accounts for situations where the developer interacts with content on a page, such as live demo widgets. Clicks that are part of text highlighting are excluded.	Strong	20 for each triggering
Cursor hovering	People tend to use the cursor to guide their attention while reading web pages [18, 52, 57, 65, 103].	Triggers each time when the mouse cursor hovers over a content block b for at least 2 seconds. This accounts for situations where the developer naturally moves the mouse cursor onto the content that is currently being read to guide his or her attention [18, 64, 102, 103]. However, a cursor hover triggering will be disqualified when the system detects an extended period of idling (2 minutes) without any user actions.	Weak	$0.5t$, where t is the duration (measured in seconds) of the cursor's stay within the bounds of content block b. The maximum score is 10. In our pilot testing, users rarely spend more than 10 seconds reading a text block.
Content dwelling	The longer some content stays visible, the more likely that the user is interested in it [22, 65].	Triggers each time when a content block b gets scrolled into and stays in the visible view port for at least 2 seconds. This indicates that the developer has at least paid attention to b. However, a dwell triggering during idling is disqualified.	Weak	$0.2t$, where t is the duration (measured in seconds) of content block b's stay in the visible browser viewport. The maximum score is 4. In our pilot testing, users rarely stay at one location for more than 10 seconds.

Table 1: Implicit behavioral signals used in Crystalline to track user attention. Column 1 lists the implicit signals; column 2 provides evidence from selected prior research on the efficacy of the signals; column 3 describes how the signals are used in Crystalline; column 4 indicates the relative strength of a signal in terms of predicting user attention; column 5 details the scoring function used to translate signal triggerings into numeric scores based on the relative signal strengths. The scoring functions were empirically determined through iterative pilot testing.

4.2.2 Organizing and prioritizing information. Not all options or criteria are equally useful to a particular developer. Prior work has suggested that a programming decision usually comes down to how well each option matches the developer's goals and criteria that he or she deemed important [42, 77, 78, 82, 92, 94, 100, 107]. In this work, we explore using the amount of attention that one pays to a particular criterion to approximate its perceived value or importance. To operationalize this, for each web page that a developer visits, Crystalline processes all the content blocks (HTML block-level elements, such as <p>, <li>, <pre>, and , etc.) to detect what options and criteria are associated with each block. Specifically, it prioritizes verbatim mentioning of options and criteria within a block, then possible options and criteria identified from section headers above the block, then web page titles. If no options are detected, the page title is used as a placeholder.

Next, Crystalline tracks each triggering of five implicit behavioral signals (*copying content*, *text highlighting*, *clicking*, *cursor hovering*, and *content dwelling*) listed in Table 1 on any content block and translates it into a numeric score (using column 5). The final attention score A_c representing the amount of attention that a user pays to a particular criterion c is then calculated using equation (1):

$$A_c = \sum_{t \in T} I(t, c) \times W(t) \tag{1}$$

where T is the set of all implicit signal triggerings; t is a particular triggering; $I(t, c)$ returns 1 if t was triggered on a content block that is associated with the criterion c, and returns 0 otherwise; and $W(t)$ is the corresponding scoring function found in the last column in Table 1. The scoring functions were empirically determined through iterative pilot testing.

To accommodate various behavioral patterns exhibited by different users, we iteratively recruited four batches of participants with diverse backgrounds and job responsibilities both within our lab and externally. We followed a diary study approach [101] by monitoring their online searching and browsing behavior related to programming through a custom chrome extension that logs triggerings of the above behavior signals and ranks the importance of the associated content blocks accordingly (the initial score functions

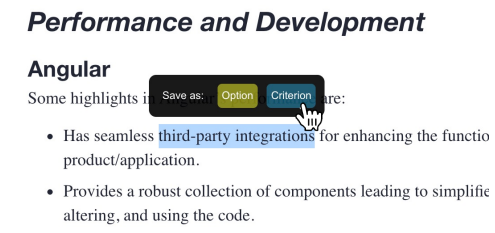

Figure 4: Using the selection popup menu to manually collect options and criteria.

were determined through our heuristics). At the end of each sensemaking episode, we prompted them to review how well the system did in inferring what they thought was important, and tuned the score function heuristics accordingly (favoring recall over precision). We leave more advanced and adaptive scoring models for future work to investigate.

By default, the system shows the top 15 criteria ranked by decreasing attention scores in both the list and the table view. Users can use the "See More" and "See Less" buttons to adjust how many criteria that they would like to see at the same time (Figure 1g). As the user browses more content and spreads his or her attention on different content blocks, the order of these criteria changes accordingly in real-time, which provides the user with an ambient awareness of what the system thinks are important. To provide users with the flexibility to override the system's ranking, they can right-click on a criterion and use the "pin this criterion" feature to pin it at the top (Figure 1i). They can additionally specify their own order of preferences by dragging and dropping to reorder the criteria in the table view, which will automatically pin a criterion if it is not already pinned. Each time an implicit behavioral signal triggering is detected, Crystalline also collects the target content block as an evidence snippet, which is presented with its original styling [81] in the detail views and the comparison table view as mentioned above.

4.2.3 Managing connections and relationships. One way for Crystalline to actively manage the relationships among the collected information is to automatically merge similar criteria together into *criteria groups* (indicated by a "multiple items" icon at the end, see Figure 1e). To achieve this, we leverage recent advances in transformer machine learning models such as Universal Sentence Encoder [13] and BERT [28] that can encode textual content into semantically meaningful vector representations called embeddings [43], i.e., two or more semantically close pieces of content will also be close in the embedding vector space (measured by a distance metric, e.g., the cosine similarity distance between vectors [113]). Crystalline computes an embedding for every criterion as the average of its own embedding and its corresponding evidence snippet, and automatically merges criteria that are within a specified semantic distance threshold to each other into a group. For example, as shown in Figure 3a, the system automatically merges "Right to Left" (taken from the option "Splide") and "RTL" (taken from the option "Swiper") together since they are semantically similar. The distance threshold was determined empirically through iterative pilot testing. This has the benefit of reducing clutter while helping users make connections among the information that they have seen, which is reported by prior work as one of the difficult steps during sensemaking and schematization [37, 96, 106]. In case the system fails to automatically group similar criteria together, users can use drag and drop to manually make the grouping. Similarly, users can easily split a criteria group by right-clicking on the group and hitting the "split this criteria group" menu item.

In situations where a user reads and investigates some criterion at one location, Crystalline will also actively look for evidence for the same or similar criteria from other pages that the user has visited (including the current page) but has not (yet) paid attention to according to the implicit signals. Crystalline will remind the user of the existence of this additional evidence through a red notification dot at the top right of a criterion (Figure 1f) as well as in the detailed views (Figure 3d). This then serves as an additional way for the system to help users uncover and manage unseen relationships among the information space, as well as a springboard for users to jump directly to the "overlooked" information for further investigation.

4.3 Implementation Notes

The Crystalline Chrome browser extension is implemented in HTML, JavaScript, and CSS, using the React JavaScript library [34]. It also uses Google's Firebase for database synchronization and persistence, back-end functions, and user authentication.

To produce the content embeddings, we used *bert-as-a-service* [28] and the `uncased_L-12_H-768_A-12` pre-trained BERT model to implement a REST API that the extension can query on-demand. The embedding calculations are known to incur significant computational costs and delays. Therefore, to ensure a smooth user experience, they are better suited to run on a remote server with the necessary resources rather than locally in an end-user's browser.

Unlike other systems [33, 95] that help users find more information from new sources, Crystalline only collects information from the web pages that a user has explicitly visited. This is an intentional design choice we make in the current implementation: the major role of Crystalline is to remove the burden for users to actively keep track of relevant information that they have personally seen and investigated so that it is easier for them to revisit and recall. We leave the design space of automating the discovery of new relevant information for future research to explore.

5 EVALUATION

We conducted an initial lab study to evaluate the usability of the Crystalline system in helping developers collect and organize information.

5.1 Participants

We recruited 12 participants (7 male, 5 female) aged 22-35 (μ = 27.6, σ = 3.7) years old through emails and social media. The participants were required to be 18 or older, fluent in English, and experienced in programming. Participants had on average 6.9 years of programming experience, with half of them currently working or having worked as a professional developer and the rest having programming experience in universities.

	Manually select information and capture	Rename an option / criteria	Delete an option / criteria	Manually put information snippets into the table	Remove a snippet from the table	Merge criteria into groups	Split criteria groups	Pin or reorder criteria	**Overall**
Task A	27.0 (6.42)	1.67 (1.97)	0.67 (1.03)	16.5 (5.43)	0.50 (0.84)	N/A	N/A	6.00 (2.19)	52.3 (13.7)
Task B	26.2 (5.56)	1.83 (1.60)	1.50 (1.38)	14.5 (5.28)	0.33 (0.82)	N/A	N/A	6.00 (1.79)	50.3 (14.3)
Average	26.6 (5.74)	1.75 (1.71)	1.08 (1.24)	15.5 (5.21)	0.42 (0.79)	N/A	N/A	6.00 (1.91)	51.3 (13.4)

(a) Unakite condition

	Manually select information and capture	Rename an option / criteria	Delete an option / criteria	Manually put information snippets into the table	Remove a snippet from the table	Merge criteria into groups	Split criteria groups	Pin or reorder criteria	**Overall**
Task A	0.83 (0.75)	2.17 (1.17)	0.50 (0.84)	0.17 (0.41)	0.33 (0.52)	2.33 (0.82)	0.83 (0.75)	5.33 (1.97)	12.5 (3.02)
Task B	1.00 (1.26)	1.67 (0.82)	0.50 (0.55)	0.33 (0.52)	0.33 (0.52)	1.83 (0.75)	0.67 (0.82)	5.50 (2.74)	11.8 (3.31)
Average	0.92 (1.00)	1.92 (1.00)	0.50 (0.67)	0.25 (0.45)	0.33 (0.49)	2.08 (0.79)	0.75 (0.75)	5.42 (2.27)	12.2 (3.04)

(b) Crystalline condition

Table 2: Statistics for the average number of interactions performed by users to perform the tasks in the user study. Standard deviations are included in the parentheses.

5.2 Procedure

The study was a within-subjects design, where participants were presented with two tasks and were asked to complete one of them using Unakite (baseline condition) and the other using Crystalline (experimental condition), in a counterbalanced order. For each task, participants were presented a programming decision-making problem, a set of four web pages, some necessary background of the problem, and a list of three options available to solve the problem that they were required to investigate. The provided web pages were either documentation pages of specific options or comprehensive review articles reviewing several options together. Participants were instructed to read through the provided web pages, and use either Unakite or Crystalline to collect and organize information into a comparison table containing all the given options and at least 8 different criteria in the order of their perceived importance. We imposed a 20-minute limit per task to keep participants from getting caught up in one of the tasks. However, they were instructed to inform the researcher when they have collected 8 criteria as well as the associated evidence. If they wished to continue beyond this checkpoint, they were allowed to, until they felt like they could make no further progress. Specifically, the two tasks were to use the corresponding system in each condition to build a comparison table of:

- *(A)* Choosing a JavaScript carousel library to build a photo sharing web application. The available options were: Splide.js[3], Slick[4], and Swiper[5].
- *(B)* Choosing a front-end framework to implement a basic personal portfolio website. The available options were: React.js[6], Angular[7], and Vue.js[8].

We chose Unakite over other commercially available tools such as Google Docs as the baseline condition because: 1) it can be easily used to capture richer contexts such as formatted text (example code), images, and links; 2) similar to Crystalline, it also provides a sidebar that allows participants to view and organize the collected information directly rather than switching context over to another browser tab or application to paste in and structure information; and 3) Unakite was shown to be easy to learn and use in prior research and incurs significantly less overhead cost than using Google Docs [81].

In addition, rather than letting participants search for their own pages to research, we provided them with the predefined set of pages to ensure a fair comparison of the results, and since helping to find relevant web pages is not a goal of Crystalline. Requiring participants to only read the predefined pages (each contains on average 7 screenfuls of content) also helps ensure that the two tasks are of roughly equal difficulty in terms of reading and cognitive processing effort. Furthermore, to ensure realism and participant engagement, the tasks were selected based on actual questions asked and discussed on programming forums and websites. We specifically simplified the requirements and background of task B to match that of task A, since otherwise, choosing a JavaScript framework (e.g., to build interactive industry-level web applications) would arguably be more substantial and involve deeper and much more careful comparisons and team discussions that are beyond the scope of this lab study. In fact, as shown in section 6.1 there was no significant difference by task.

Each study session started by obtaining consent and having participants fill out a demographic survey. Participants were then given a 10-minute tutorial showcasing the various features of Unakite and Crystalline and a 10-minute practice session on both systems before starting. At the end of the study, the researcher conducted a survey and an interview eliciting subjective feedback on the Unakite and Crystalline experience. Each study session took approximately 60 minutes, using a designated MacBook Pro computer with Chrome, Unakite and Crystalline installed. All sessions were carried out in person, with participants and the researcher appropriately masked following COVID-19 mitigation protocols. All participants were compensated $15 for their time. The study was approved by our institution's IRB office.

[3]https://splidejs.com/
[4]https://kenwheeler.github.io/slick/
[5]https://swiperjs.com/
[6]https://reactjs.org/
[7]https://angular.io/
[8]https://vuejs.org/

Question Statements	Crystalline condition	Unakite condition
I would consider my interactions with the tool to be understandable and clear.	6.17 (0.39)	6.08 (0.67)
I would consider it easy for me to learn how to use this tool.	6.08 (0.79)	6.00 (1.04)
I enjoyed the features provided by the tool.	6.25 (0.45)	6.17 (0.58)
Using this tool would make solving programming problems at my work more efficient and effective.	6.08 (0.29)*	5.75 (0.45)*
If possible, I would recommend the tool to my friends and colleagues doing programming work.	6.17 (0.58)*	5.58 (0.51)*

Table 3: Statistics of scores in the post-tasks survey. Participants were asked to rate their agreement with statements related to their experience interacting with Crystalline and Unakite on a 7-point Likert scale from "Strongly Disagree" (a score of 1) to "Strongly Agree" (a score of 7). Statistics in column 2 and 3 are presented in the form of mean (standard deviation). Statistically significant differences ($p < 0.05$) through paired t-tests are marked with an *.

6 RESULTS

6.1 Quantitative Results

All participants were able to complete all of the tasks in both conditions, and nobody went over the pre-imposed time limit. Figure 1, together with Figure 3, shows an example table built by one of the participants in the study for task A.

To examine how Crystalline performs compared to the baseline Unakite condition, we measured the time it took for participants to finish each task. A two-way repeated measures ANOVA was conducted to examine the within-subject effects of condition (Crystalline vs. Unakite) and task (A vs. B) on task completion time. There was a statistically significant effect of condition ($F(1, 20) = 8.06$, $p = 0.01$) such that participants completed tasks significantly faster (21.6% faster) with Crystalline (Mean = 611.8 seconds, SD = 144.6 seconds) than in the Unakite condition (Mean = 780.3 seconds, SD = 137.6 seconds). There was no significant effect of task ($F(1, 20) = 0.11$, $p = 0.74$), indicating the two tasks were indeed of roughly equal difficulty. These results suggest Crystalline helped participants build up comparison tables faster overall, even the majority of their time was necessarily spent reading through the material in both conditions.

To account for this reading time, we also compared the *overhead cost* [81] of using both tools to collect and organize information. For the Crystalline condition, we calculated the overhead cost as the portion of the time participants spent on directly interacting with Crystalline (scrolling through the list and table view to examine the evidence collected so far, splitting and merging criteria, pinning important criteria, manually collecting information, etc.) out of the total time they used for a task (vs. reading and comprehending the web pages). Similarly, in the Unakite condition, the overhead cost was calculated as the percent of time participants spent on directly using Unakite features (selecting and collecting information snippets, drag and dropping snippets into the comparison table, etc.), in the same way as was done to compare Unakite to Google Docs [81].

A two-way repeated measures ANOVA was conducted to examine the within-subject effects of condition (Crystalline vs. Unakite) and task (A vs. B) on overhead cost. There was a statistically significant effect of condition ($F(1, 20) = 77.5$, $p < 0.001$) such that the overhead cost was significantly lower (almost 60% lower) in the Crystalline condition (Mean = 11.6%, SD = 0.04) than in the Unakite condition (Mean = 28.4%, SD = 0.07). Again, there was no significant effect of task ($F(1, 20) = 0.53$, $p = 0.48$)). Thus, using Crystalline resulted in reduced overhead costs of collecting and organizing information.

To gain deeper insights into *why* the overhead cost was significantly lower in the Crystalline condition, we tallied the number of interactions performed in each task while collecting and organizing information to build the comparison tables (Table 2). Here, we notice that the majority of interactions in the Unakite condition are to manually collect information snippets (on average 26.6 times) and place them into the comparison table (on average 15.5 times). In contrast, in the Crystalline condition, the majority of interactions are to merge criteria into groups (on average 2.08 times) and pin or reorder the criteria in the table (on average 5.42 times). This suggests that, to some extent, Crystalline has transformed the previously active capturing and organizing work into passive monitoring and error-fixing, which explains the lower overhead cost.

In the survey, participants reported (in 7-point Likert scales) that they thought the interactions with Crystalline were understandable and clear (Mean = 6.17, SD = 0.39), Crystalline was easy to learn (Mean = 6.08, SD = 0.79), and they enjoyed Crystalline's features (Mean = 6.25, SD = 0.45). In addition, compared to Unakite (Mean = 5.75, SD = 0.45), they thought using Crystalline (Mean = 6.08, SD = 0.29) would help them solve programming problems more efficiently and effectively, and would recommend Crystalline (Mean = 6.17, SD = 0.58) over Unakite (Mean = 5.58, SD = 0.51) to friends and colleagues doing programming work, both differences were statistically significant under paired t-tests. Details of the survey questions and scores are presented in Table 3.

6.2 Qualitative Observations

6.2.1 Usability and usage patterns. Overall, participants appreciated the increased efficiency afforded by various Crystalline features. Many (9/12) mentioned that the perceived workload to collect and organize what they have investigated was minimal, saying that "*I feel like I got a table for free*" (P3), "*the fact that I can see what I've paid a lot of attention to automatically bubbles up to the top is quite magical*" (P9), and "*It feels as if I was sitting in the passenger seat and not having to do all the steering and maneuvering*" (P7). Some (3/12) participants also reported having taken advantage of the overlooked information reminder feature (Figure 3d) to guide their research. Furthermore, participants reflected that Crystalline

relieves them of the burden of trying to anticipate the value of a particular piece of information before collecting it since "*the important bits will eventually be at or near the top, hopefully*" (P12), and they could "*focus on reading the page itself and not context switch to bookkeeping mode again and again*" (P5).

However, some did voice concerns about the system's ability at the beginning of the tasks, arguing that they were "*skeptical if it will actually collect the right things*" (P1), and reported that they would "*skim through the list view and the table view quite frequently at the beginning*" (P7). However, as they progressed through the tasks, their confidence in Crystalline increased, and they only occasionally checked the sidebar. We observed that three of the 12 participants ended up not examining and editing the system's output until they felt like they had finished reading and processing all the given pages, and they made minimal edits to the results.

6.2.2 Working with machine suggestions. Participants generally thought that the benefits of automating the collection and organization process outweighed the costs of dealing with occasional unhelpful machine suggestions, such as incorrectly merging criteria together or prioritizing unimportant criteria at the top of the list. For example, P7 reflected, "*it feels like a mind reader. I know it's not perfect, but I also don't expect it to be, and would actually prefer occasionally peeking into what it's been doing and fixing whatever that's not correct than grabbing everything by myself all the time.*"

Some did raise concerns about the ordering of criteria getting changed too frequently ("*they [the criteria] were jumping around*", P7) at the beginning. This is likely due to the fact that users were skimming through a web page without paying particular attention to anything at the beginning, causing their attention scores to be relatively indistinguishable. For future iterations of the system, we could experiment with less frequent UI update intervals under these circumstances so it would cause less distraction.

6.3 Evaluation Discussion

Similar to what was reported in prior work [99], since our participants were not explicitly told how the system worked to automatically collect and rank information, they had to form their own mental models and hypotheses about how the system works and how they could affect it with their behavior. For example, P8 noticed that "*it looks like if I spend a little bit more time on a particular place on a page, the corresponding criterion would get picked up and bumped up quickly; and if I click on that part a bunch of times, which happens to be what I typically would do when I try to focus my attention on something now that I'm thinking about it, it's [the corresponding criterion] going to go up even faster.*" This suggests that our implicit signals were working, and further, that with experience users might adapt to *explicitly* steer the system towards their goal of collecting and prioritizing information, resulting in, to some extent, a mixed-initiative collection approach that still would require much less effort than the baseline methods. Future research could explore the costs and benefits of a wide variety of interactions and signals that lie on the spectrum between implicit behavioral signals to full manual direct manipulations, and any differences caused by directly instructing users about the implicit signals being used.

Though the current version of Crystalline mainly focuses on reducing the cost for developers to collect and organize information, which was exactly what we tested in the lab study, we were also interested in making sure that the *quality* of the comparison tables built using Crystalline does not degrade as seen in other automation scenarios [49, 111]. Since there is not a gold standard comparison table, we evaluated the correctness of Crystalline's automatic approaches by how much editing participants had to do in order to fix Crystalline's mistakes and make sure that all the content in the table was eventually filled out and ranked correctly according to their understanding as per the study protocol. As shown in Table 2(b), participants only had to perform on average 12.2 edits to the automatically generated comparison tables, compared to the 51.3 actions that they had to manually perform in the baseline Unakite condition (the difference is statistically significant, $p < 0.01$). Among these, edits that are related to collecting information, such as manually selecting information and capture (0.92 times), renaming (1.92 times), and deleting information (0.50 times) were minimal, suggesting that our combination of NLP and behavioral signal heuristics was working effectively to collect information that the users thought was important. However, participants pinned or reordered the criteria that were automatically ranked by Crystalline on average 5.42 times (SD = 2.27 times). One possible explanation is that the universal scoring functions (in Table 1) did not necessarily apply to every single participant, suggesting the need for a more sophisticated and personalized scoring mechanism in future iterations of Crystalline and systems that leverage signals from users' natural browsing behavior.

In addition, we asked and coded their opinions about *using* these tables as if they were the subsequent developers trying to *understand* the design rationale. In general, participants were excited about using comparison tables automatically built by Crystalline. For example, P10 highlighted scenarios where Crystalline would be useful for his own purposes, saying that "*it's sort of like a never-erased whiteboard that would most likely help me remember what I looked at three months ago.*" In addition, some reflected that compared to having no clue of why a decision was made in a particular way in the first place, they would appreciate at least having access to a Crystalline table even if it was not actively monitored and maintained during the initial developer's sensemaking process. For example, P4 said: "*I think being able to read something like this [Crystalline table] is going to make a big difference when you're banging your head against the wall trying to understand why this particularly old API was chosen, I mean, especially when the guy who wrote the code was long gone, I could at least 'read a transcription of his mind' in some sense.*" Here, we see preliminary evidence that our approach of automatically collecting and organizing information on behalf of developers is useful and valuable. We leave the formal evaluation of the quality of fully automatically built comparison tables with possibly more advanced versions of Crystalline for future work.

7 LIMITATIONS

Currently, Crystalline works best on a limited set of web pages in the programming domain, including documentation pages that are dedicated to a particular library or a set of APIs, as well as review articles or question answering pages that discuss and compare several

options together. We chose to optimize for these types of web pages in the current prototype as they are reported in prior work [63, 81] as well as our formative discussions with developers as some of the most frequently consulted programming resources when it comes to making decisions. However, the performance reported on the web pages used in the study is not necessarily representative of how Crystalline would operate even on web pages of these types for users in general. In addition, Crystalline currently relies heavily on the overall structure of the web pages being standard, meaning that a page uses HTML tags appropriately according to their semantics (e.g., enclosing headers and list items in `<h>` and `<li>` tags rather than wrapping everything with `<div>` tags) and that there is a strong semantic coherence between a section header and its corresponding content. Though this is sufficient to demonstrate the idea of automatic collecting and organization and the benefits they offer, future research is needed to make Crystalline-style tools work on a more diverse set of web pages, as well as how to be clear upfront about its limitations in parsing web pages that do not follow appropriate web standards.

Furthermore, our lab study has several limitations. Given the short amount of training and practice time participants had, some might not have been able to fully grasp the various features of Crystalline, or they might have been confused about what Unakite (the baseline system) has to offer. The study tasks might not be what participants typically encounter in their daily work, depending on whether they are in a position to make decisions, and thus they may not be equipped with the necessary motivation or context that they would otherwise have in real life. We mitigate these risks in the study setup by: 1) having participants perform a practice task for each condition simulating what they would have to do in the real tasks; 2) choosing the study tasks based on actual questions that are discussed by developers on Stack Overflow and other popular programming community forums; and 3) providing participants with sufficient background information and context to help them get prepared. In fact, 7 out of 12 participants reported that the tasks were indeed similar to what they would deal with in their daily work. We would like to further address these limitations in the future by having developers use Crystalline on their own work and personal projects, which would provide them with sufficient motivation as well as experience with Crystalline enriched over time.

Finally, the overhead cost measurement in the study could be conservative, as we did not account for the time participants spent simply glancing or looking at the sidebars without any explicit interactions with it. However, from our observations during the study, participants rarely spent any extended time doing this. Nevertheless, we would like to take advantage of more advanced tools such as eye tracking [7, 89, 90, 103] in the future to more accurately account for the proportion of time when a participant's gaze is fixated on the user interface of the tools rather than on actual web content.

8 FUTURE WORK

Through designing and evaluating Crystalline, we gained deeper insights into the benefits and trade-offs of automatically collecting and organizing information for developers as they make sense of the web to make programming decisions. This motivates some ideas for future work.

While Crystalline's approach provides developers with an inexpensive way of capturing knowledge in the browser, it represents only one piece of a larger puzzle of how to support a developer's everyday work that involves sensemaking and decision making. One dimension to characterize this is that developers also frequently perform activities outside their browsers, such as in IDEs, code editors [108], command-line interfaces [19], literate programming notebooks [68, 69], or threads of discussions during formal or informal meetings [125]. Further research would be needed to understand how to collect and organize information from these sources as well as how to integrate them together to provide a more comprehensive picture of the decision making context. Another dimension that is relevant is the lifecycle of the knowledge captured via systems like Unakite and Crystalline. Early evidence from the user study has suggested there is a benefit of Crystalline's organization from the perspective of a subsequent developer who may need to understand a previous developer's decision. Future research could investigate how well developers are able to understand and potentially reuse these automatically assembled knowledge artifacts, possibly without any manual interventions from the initial knowledge authors, which could, in turn, eliminate the starting cost associated with initial knowledge creation [37] and unlock the virtuous cycle of accelerated programming knowledge reuse [37, 82].

Though the current set of mechanisms for deriving the importance of criteria from implicit behavioral signals generally works well for the setting of this research, there could be situations where a user's default browsing behaviors and patterns fall outside the limited set of signals and heuristics that Crystalline is currently looking for. For example, a user might not have the habit of unconsciously using the cursor as a reading guide or might not interact with the page at all while reading, which would render the tracking of some of the behavioral signals moot. In addition, users could exhibit different or additional behavior patterns when generalized to other tasks domains that involve information-backed decision making, such as comparison shopping, trip-planning, etc. [15, 53]. For example, when interacting with a map view to find the best local dining option, a user may frequently pan around and zoom (in and out) to view different restaurants, and both the duration of stay on a particular restaurant and how many times it is viewed back and forth could be leveraged to approximate the user's interest and investment of effort. One way to address these concerns is to leverage a more diverse set of behavioral signals and potentially signal combinations, such as scrolling, mouse panning, zooming, eye tracking [35, 36, 89, 90], and facial gestures tracking [70, 117] to collect a more accurate picture of what users are seeing on screen. Another future direction that could be fruitful is to take a machine learning approach instead of the current rule-based approach for approximating content importance using behavioral signals. Specifically, we could leverage recent advances in crowdsourcing and labeling [17, 20, 27, 114] to log, annotate, and construct a large-scale data set that maps a variety of behavioral signals to the perceived importance of content blocks that they are triggered on, and train on this data set to obtain scoring functions that would work more widely. Alternatively, an online learning approach could also be promising, where the system continuously learns, adapts, and

improves from an individual user's behavior over time, as suggested by Horvitz [62].

Last but not least, automation afforded by systems like Crystalline enable people to focus their attention on reading and comprehending the web pages rather than splitting attention with having to collect and organize the information at the same time. However, prior work in learning science, such as Bransford et al. [24], found that people who *personally* performed the actions of collecting, categorizing, and organizing information were more likely to be able to recall it correctly and in detail, and exhibited increased confidence in the final outcome. This raises an interesting tension and trade-off between full-on automation and direct manipulation — future research would be required to examine the long term effect on people's learning outcome as well as confidence in their decisions using systems like Crystalline, and determine the appropriate levels and circumstances when automatic information bookkeeping should be applied.

9 CONCLUSION

This paper explored how automatically collecting and organizing information as developers search and browse the web can better support them in decision making scenarios. Our designs were motivated by the growing complexity of the decisions that developers need to make, and the lack of tooling support to help them efficiently gather and synthesize evidence without causing much interruption to their main focus of reading and understanding content online. We introduced Crystalline, a browser extension that instantiates this idea by leveraging natural language processing and users' behavior signals such as mouse movement and dwell time to infer what information to collect and how to organize and prioritize it on behalf of a user. Through a lab study with 12 participants, we found promising evidence that using Crystalline as a copilot to collect and organize information is much faster and more efficient, and the resulting knowledge artifacts are potentially useful and valuable for the initial user as well as for subsequent consumption by people who need to understand the original decision-making context.

ACKNOWLEDGMENTS

This research was supported in part by NSF grants CCF-1814826 and FW-HTF-RL-1928631, Google, Bosch, the Office of Naval Research, and the CMU Center for Knowledge Acceleration. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors. We would like to thank our study participants for their kind participation and our anonymous reviewers for their insightful feedback. We are genuinely grateful to Yongsung Kim, Joseph Chee Chang, and Amber Horvath for their valuable feedback. In addition, we sincerely thank Jinlei Chen, Tianying Chen, Yulan Feng, Nan Gao, Haojian Jin, Toby Jia-Jun Li, Franklin Mingzhe Li, Julia Jiayin Qian, Haitian Sun, Jiachen Wang, Eric Yiyi Wang, Ziyan Wang, Zheng Yao, and Yi Zhou for their constant support, especially during the COVID-19 pandemic.

REFERENCES

[1] 2012. Google Notebook. https://www.google.com/googlenotebook/faq.html

[2] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N. Bennett, Kori Inkpen, Jaime Teevan, Ruth Kikin-Gil, and Eric Horvitz. 2019. Guidelines for Human-AI Interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. ACM, New York, NY, USA, 3:1–3:13. https://doi.org/10.1145/3290605.3300233 event-place: Glasgow, Scotland Uk.

[3] Michelle Q. Wang Baldonado and Terry Winograd. 1997. SenseMaker: An Information-exploration Interface Supporting the Contextual Evolution of a User's Interests. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI '97)*. ACM, New York, NY, USA, 11–18. https://doi.org/10.1145/258549.258563

[4] David Bawden, Clive Holtham, and Nigel Courtney. 1999. Perspectives on information overload. *Aslib Proceedings* 51, 8 (Jan. 1999), 249–255. https://doi.org/10.1108/EUM0000000006984 Publisher: MCB UP Ltd.

[5] Krishna Bharat. 2000. SearchPad: explicit capture of search context to support Web search. *Computer Networks* 33, 1 (June 2000), 493–501. https://doi.org/10.1016/S1389-1286(00)00047-5

[6] Eric A. Bier, Edward W. Ishak, and Ed Chi. 2006. Entity quick click: rapid text copying based on automatic entity extraction. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems (CHI EA '06)*. Association for Computing Machinery, New York, NY, USA, 562–567. https://doi.org/10.1145/1125451.1125570

[7] Jeffrey P. Bigham, Mingzhe Li, Samuel C. White, Xiaoyi Zhang, Qi Shan, and Carlos E. GUESTRIN. 2021. On-the-fly calibration for improved on-device eye tracking. https://patents.google.com/patent/US11106280B1/en

[8] Joel Brandt, Mira Dontcheva, Marcos Weskamp, and Scott R. Klemmer. 2010. Example-centric Programming: Integrating Web Search into the Development Environment. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. ACM, New York, NY, USA, 513–522. https://doi.org/10.1145/1753326.1753402

[9] Joel Brandt, Philip J. Guo, Joel Lewenstein, Mira Dontcheva, and Scott R. Klemmer. 2009. Two Studies of Opportunistic Programming: Interleaving Web Foraging, Learning, and Writing Code. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 1589–1598. https://doi.org/10.1145/1518701.1518944 event-place: Boston, MA, USA.

[10] Georg Buscher, Edward Cutrell, and Meredith Ringel Morris. 2009. What do you see when you're surfing? using eye tracking to predict salient regions of web pages. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 21–30. https://doi.org/10.1145/1518701.1518705

[11] Georg Buscher, Andreas Dengel, and Ludger van Elst. 2008. Eye movements as implicit relevance feedback. In *CHI '08 Extended Abstracts on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 2991–2996. https://doi.org/10.1145/1358628.1358796

[12] Georg Buscher, Ludger van Elst, and Andreas Dengel. 2009. Segment-level display time as implicit feedback: a comparison to eye tracking. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval (SIGIR '09)*. Association for Computing Machinery, New York, NY, USA, 67–74. https://doi.org/10.1145/1571941.1571955

[13] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal Sentence Encoder. *arXiv:1803.11175 [cs]* (April 2018). http://arxiv.org/abs/1803.11175 arXiv: 1803.11175.

[14] Joseph Chee Chang, Nathan Hahn, and Aniket Kittur. 2016. Supporting Mobile Sensemaking Through Intentionally Uncertain Highlighting. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16)*. ACM, New York, NY, USA, 61–68. https://doi.org/10.1145/2984511.2984538

[15] Joseph Chee Chang, Nathan Hahn, and Aniket Kittur. 2020. Mesh: Scaffolding Comparison Tables for Online Decision Making. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology (UIST '20)*. Association for Computing Machinery, New York, NY, USA, 391–405. https://doi.org/10.1145/3379337.3415865

[16] Joseph Chee Chang, Yongsung Kim, Victor Miller, Michael Xieyang Liu, Brad A Myers, and Aniket Kittur. 2021. Tabs.do: Task-Centric Browser Tab Management. In *The 34th Annual ACM Symposium on User Interface Software and Technology*. Association for Computing Machinery, New York, NY, USA, 663–676. https://doi.org/10.1145/3472749.3474777

[17] Yu-Wei Chao, Yunfan Liu, Xieyang Liu, Huayi Zeng, and Jia Deng. 2018. Learning to Detect Human-Object Interactions. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 381–389. https://doi.org/10.1109/WACV.2018.00048

[18] Mon Chu Chen, John R. Anderson, and Myeong Ho Sohn. 2001. What can a mouse cursor tell us more? correlation of eye/mouse movements on web browsing. In *CHI '01 Extended Abstracts on Human Factors in Computing Systems (CHI EA '01)*. Association for Computing Machinery, New York, NY, USA, 281–282. https://doi.org/10.1145/634067.634234

[19] Yan Chen, Jaylin Herskovitz, Walter S. Lasecki, and Steve Oney. 2020. Bashon: A Hybrid Crowd-Machine Workflow for Shell Command Synthesis. In *2020 IEEE*

Symposium on Visual Languages and Human-Centric Computing (VL/HCC). 1–8. https://doi.org/10.1109/VL/HCC50065.2020.9127248 ISSN: 1943-6106.

[20] Yan Chen, Maulishree Pandey, Jean Y. Song, Walter S. Lasecki, and Steve Oney. 2020. Improving Crowd-Supported GUI Testing with Structural Guidance. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376835

[21] Alex Chitu. 2011. Google Sets Will Be Shut Down. http://googlesystem.blogspot.com/2011/08/google-sets-will-be-shut-down.html

[22] Mark Claypool, Phong Le, Makoto Wased, and David Brown. 2001. Implicit interest indicators. In *Proceedings of the 6th international conference on Intelligent user interfaces (IUI '01)*. Association for Computing Machinery, New York, NY, USA, 33–40. https://doi.org/10.1145/359784.359836

[23] A. Cockburn and J. Highsmith. 2001. Agile software development, the people factor. *Computer* 34, 11 (Nov. 2001), 131–133. https://doi.org/10.1109/2.963450 Conference Name: Computer.

[24] National Research Council and others. 2000. *How people learn: Brain, mind, experience, and school: Expanded edition*. National Academies Press.

[25] Sergio Cozzetti B. de Souza, Nicolas Anquetil, and Káthia M. de Oliveira. 2005. A study of the documentation essential to software maintenance. In *Proceedings of the 23rd annual international conference on Design of communication: documenting & designing for pervasive information (SIGDOC '05)*. Association for Computing Machinery, New York, NY, USA, 68–75. https://doi.org/10.1145/1085313.1085331

[26] R. DeLine, M. Czerwinski, and G. Robertson. 2005. Easing Program Comprehension by Sharing Navigation Data. In *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05)*. IEEE, 241–248. https://doi.org/10.1109/VLHCC.2005.32

[27] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 248–255. https://doi.org/10.1109/CVPR.2009.5206848 ISSN: 1063-6919.

[28] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]* (May 2019). http://arxiv.org/abs/1810.04805 arXiv: 1810.04805.

[29] Mira Dontcheva, Steven M. Drucker, Geraldine Wade, David Salesin, and Michael F. Cohen. 2006. Summarizing Personal Web Browsing Sessions. In *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology (UIST '06)*. ACM, New York, NY, USA, 115–124. https://doi.org/10.1145/1166253.1166273

[30] Tore Dybå and Torgeir Dingsøyr. 2008. Empirical studies of agile software development: A systematic review. *Information and Software Technology* 50, 9 (Aug. 2008), 833–859. https://doi.org/10.1016/j.infsof.2008.01.006

[31] Dora Dzvonyar, Stephan Krusche, Rana Alkadhi, and Bernd Bruegge. 2016. Context-Aware User Feedback in Continuous Software Evolution. In *2016 IEEE/ACM International Workshop on Continuous Software Evolution and Delivery (CSED)*. 12–18. https://doi.org/10.1109/CSED.2016.011

[32] Neil A. Ernst, Stephany Bellomo, Ipek Ozkaya, Robert L. Nord, and Ian Gorton. 2015. Measure it? Manage it? Ignore it? software practitioners and technical debt. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2015)*. Association for Computing Machinery, New York, NY, USA, 50–60. https://doi.org/10.1145/2786805.2786848

[33] Evernote. [n. d.]. Best Note Taking App - Organize Your Notes with Evernote. https://evernote.com

[34] Facebook. 2018. React - A JavaScript library for building user interfaces. https://reactjs.org/

[35] Mingming Fan, Zhen Li, and Franklin Mingzhe Li. 2020. Eyelid Gestures on Mobile Devices for People with Motor Impairments. In *The 22nd International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '20)*. Association for Computing Machinery, New York, NY, USA, 1–8. https://doi.org/10.1145/3373625.3416987

[36] Mingming Fan, Zhen Li, and Franklin Mingzhe Li. 2021. Eyelid gestures for people with motor impairments. *Commun. ACM* 65, 1 (Dec. 2021), 108–115. https://doi.org/10.1145/3498367

[37] Kristie Fisher, Scott Counts, and Aniket Kittur. 2012. Distributed Sensemaking: Improving Sensemaking by Leveraging the Efforts of Previous Users. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 247–256. https://doi.org/10.1145/2207676.2207711

[38] Beat Fluri, Michael Wursch, Emanuel Giger, and Harald C Gall. 2009. Analyzing the co-evolution of comments and source code. *Software Quality Journal* 17, 4 (Dec. 2009), 367–394.

[39] Andrew Forward and Timothy C. Lethbridge. 2002. The relevance of software documentation, tools and technologies: a survey. In *Proceedings of the 2002 ACM symposium on Document engineering (DocEng '02)*. Association for Computing Machinery, New York, NY, USA, 26–33. https://doi.org/10.1145/585058.585065

[40] David Foster. 2020. The Google 'vs' Trick. https://medium.com/applied-data-science/the-google-vs-trick-618c8fd5359f

[41] Martin Fowler. 2018. *Refactoring: improving the design of existing code*. Addison-Wesley Professional.

[42] Andreas Gizas, Sotiris Christodoulou, and Theodore Papatheodorou. 2012. Comparative Evaluation of Javascript Frameworks. In *Proceedings of the 21st International Conference on World Wide Web (WWW '12 Companion)*. ACM, New York, NY, USA, 513–514. https://doi.org/10.1145/2187980.2188103

[43] Yoav Goldberg and Omer Levy. 2014. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv:1402.3722 [cs, stat]* (Feb. 2014). http://arxiv.org/abs/1402.3722 arXiv: 1402.3722.

[44] Google. [n. d.]. Angular - Angular Routing. https://angular.io/guide/routing-overview

[45] Google. [n. d.]. Angular - Introduction to Angular animations. https://angular.io/guide/animations

[46] Google. [n. d.]. Angular - Validating form input. https://angular.io/guide/form-validation

[47] Google. 2019. Angular - One Framework. Mobile & Desktop. https://angular.io/

[48] Google. 2021. Cloud Natural Language. https://cloud.google.com/natural-language

[49] Katja Grace, John Salvatier, Allan Dafoe, Baobao Zhang, and Owain Evans. 2018. Viewpoint: When Will AI Exceed Human Performance? Evidence from AI Experts. *Journal of Artificial Intelligence Research* 62 (July 2018), 729–754. https://doi.org/10.1613/jair.1.11222

[50] Qi Guo and Eugene Agichtein. 2008. Exploring mouse movements for inferring query intent. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '08)*. Association for Computing Machinery, New York, NY, USA, 707–708. https://doi.org/10.1145/1390334.1390462

[51] Qi Guo and Eugene Agichtein. 2010. Ready to buy or just browsing? detecting web searcher goals from interaction data. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval (SIGIR '10)*. Association for Computing Machinery, New York, NY, USA, 130–137. https://doi.org/10.1145/1835449.1835473

[52] Qi Guo and Eugene Agichtein. 2010. Towards predicting web searcher gaze position from mouse movements. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 3601–3606. https://doi.org/10.1145/1753846.1754025

[53] Nathan Hahn, Joseph Chee Chang, and Aniket Kittur. 2018. Bento Browser: Complex Mobile Search Without Tabs. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, Montreal QC, Canada, 251:1–251:12. https://doi.org/10.1145/3173574.3173825

[54] Björn Hartmann, Mark Dhillon, and Matthew K. Chan. 2011. HyperSource: Bridging the Gap Between Source and Code-related Web Sites. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 2207–2210. https://doi.org/10.1145/1978942.1979263 event-place: Vancouver, BC, Canada.

[55] Andrew Head, Elena L. Glassman, Björn Hartmann, and Marti A. Hearst. 2018. Interactive Extraction of Examples from Existing Code. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3173574.3173659

[56] Tom-Michael Hesse and Barbara Paech. 2013. Supporting the collaborative development of requirements and architecture documentation. In *2013 3rd International Workshop on the Twin Peaks of Requirements and Architecture (TwinPeaks)*. 22–26. https://doi.org/10.1109/TwinPeaks-2.2013.6617355

[57] Yoshinori Hijikata. 2004. Implicit user profiling for on demand relevance feedback. In *Proceedings of the 9th international conference on Intelligent user interfaces (IUI '04)*. Association for Computing Machinery, New York, NY, USA, 198–205. https://doi.org/10.1145/964442.964480

[58] Ken Hinckley, Xiaojun Bi, Michel Pahud, and Bill Buxton. 2012. Informal Information Gathering Techniques for Active Reading. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 1893–1896. https://doi.org/10.1145/2207676.2208327 event-place: Austin, Texas, USA.

[59] Raphael Hoffmann, James Fogarty, and Daniel S. Weld. 2007. Assieme: Finding and Leveraging Implicit References in a Web Search Interface for Programmers. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology (UIST '07)*. ACM, New York, NY, USA, 13–22. https://doi.org/10.1145/1294211.1294216 event-place: Newport, Rhode Island, USA.

[60] Andrew Hogue and David Karger. 2005. Thresher: automating the unwrapping of semantic content from the World Wide Web. In *Proceedings of the 14th international conference on World Wide Web (WWW '05)*. Association for Computing Machinery, New York, NY, USA, 86–95. https://doi.org/10.1145/1060745.1060762

[61] Amber Horvath, Michael Xieyang Liu, River Hendriksen, Connor Shannon, Emma Paterson, Kazi Jawad, Andrew Macvean, and Brad Myers. 2021. Understanding How Programmers Can Use Annotations on Documentation. *arXiv:2111.08684 [cs]* (Nov. 2021). http://arxiv.org/abs/2111.08684 arXiv: 2111.08684.

[62] Eric Horvitz. 1999. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems (CHI '99)*. Association for Computing Machinery, New York, NY, USA, 159–166. https://doi.org/10.1145/302979.303030

[63] Jane Hsieh, Michael Xieyang Liu, Brad A. Myers, and Aniket Kittur. 2018. An Exploratory Study of Web Foraging to Understand and Support Programming Decisions. In *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 305–306. https://doi.org/10.1109/VLHCC.2018.8506517 ISSN: 1943-6092.

[64] Jeff Huang, Thomas Lin, and Ryen W. White. 2012. No search result left behind: branching behavior with browser tabs. In *Proceedings of the fifth ACM international conference on Web search and data mining - WSDM '12*. ACM Press, Seattle, Washington, USA, 203. https://doi.org/10.1145/2124295.2124322

[65] Jeff Huang, Ryen W. White, Georg Buscher, and Kuansan Wang. 2012. Improving searcher models using mouse cursor activity. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval (SIGIR '12)*. Association for Computing Machinery, New York, NY, USA, 195–204. https://doi.org/10.1145/2348283.2348313

[66] Zachary Ives, Craig Knoblock, Steve Minton, Marie Jacob, Partha Talukdar, Rattapoom Tuchinda, Jose Luis Ambite, Maria Muslea, and Cenk Gazen. 2009. Interactive Data Integration through Smart Copy & Paste. *arXiv:0909.1769 [cs]* (Sept. 2009). http://arxiv.org/abs/0909.1769 arXiv: 0909.1769.

[67] Harish Kandala, B. K. Tripathy, and K. Manoj Kumar. 2018. A Framework to Collect and Visualize User's Browser History for Better User Experience and Personalized Recommendations. In *Information and Communication Technology for Intelligent Systems (ICTIS 2017) - Volume 1 (Smart Innovation, Systems and Technologies)*, Suresh Chandra Satapathy and Amit Joshi (Eds.). Springer International Publishing, Cham, 218–224. https://doi.org/10.1007/978-3-319-63673-3_26

[68] Mary Beth Kery, Amber Horvath, and Brad Myers. 2017. Variolite: Supporting Exploratory Programming by Data Scientists. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 1265–1276. https://doi.org/10.1145/3025453.3025626

[69] Mary Beth Kery, Marissa Radensky, Mahima Arya, Bonnie E. John, and Brad A. Myers. 2018. The Story in the Notebook: Exploratory Data Science using a Literate Programming Tool. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–11. https://doi.org/10.1145/3173574.3173748

[70] Mohammad Kianpisheh, Franklin Mingzhe Li, and Khai N. Truong. 2019. Face Recognition Assistant for People with Visual Impairments. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 3 (Sept. 2019), 90:1–90:24. https://doi.org/10.1145/3351248

[71] Aniket Kittur, Andrew M. Peters, Abdigani Diriye, and Michael Bove. 2014. Standing on the Schemas of Giants: Socially Augmented Information Foraging. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW '14)*. ACM, New York, NY, USA, 999–1010. https://doi.org/10.1145/2531602.2531644

[72] Aniket Kittur, Andrew M. Peters, Abdigani Diriye, Trupti Telang, and Michael R. Bove. 2013. Costs and Benefits of Structured Information Foraging. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 2989–2998. https://doi.org/10.1145/2470654.2481415

[73] Amy J. Ko, Brad A. Myers, and Htet Htet Aung. 2004. Six Learning Barriers in End-User Programming Systems. In *Proceedings of the 2004 IEEE Symposium on Visual Languages - Human Centric Computing (VLHCC '04)*. IEEE Computer Society, Washington, DC, USA, 199–206. https://doi.org/10.1109/VLHCC.2004.47

[74] Amy J. Ko, Brad A. Myers, Michael J. Coblenz, and Htet Htet Aung. 2006. An Exploratory Study of How Developers Seek, Relate, and Collect Relevant Information during Software Maintenance Tasks. *IEEE Transactions on Software Engineering* 32, 12 (Dec. 2006), 971–987. https://doi.org/10.1109/TSE.2006.116

[75] Thomas D. LaToza and Brad A. Myers. 2010. Hard-to-answer Questions About Code. In *Evaluation and Usability of Programming Languages and Tools (PLATEAU '10)*. ACM, New York, NY, USA, 8:1–8:6. https://doi.org/10.1145/1937117.1937125

[76] Thomas D. LaToza, Gina Venolia, and Robert DeLine. 2006. Maintaining Mental Models: A Study of Developer Work Habits. In *Proceedings of the 28th International Conference on Software Engineering (ICSE '06)*. ACM, New York, NY, USA, 492–501. https://doi.org/10.1145/1134285.1134355

[77] John Lawrence, Jonas Malmsten, Andrey Rybka, Daniel Sabol, and Ken Triplin. 2017. Comparing TensorFlow Deep Learning Performance Using CPUs, GPUs, Local PCs and Cloud. *Publications and Research* (May 2017). https://academicworks.cuny.edu/bx_pubs/50

[78] K. Lei, Y. Ma, and Z. Tan. 2014. Performance Comparison and Evaluation of Web Development Technologies in PHP, Python, and Node.js. In *2014 IEEE 17th International Conference on Computational Science and Engineering*. 661–668. https://doi.org/10.1109/CSE.2014.142

[79] T.C. Lethbridge, J. Singer, and A. Forward. 2003. How software engineers use documentation: the state of the practice. *IEEE Software* 20, 6 (Nov. 2003), 35–39. https://doi.org/10.1109/MS.2003.1241364 Conference Name: IEEE Software.

[80] Franklin Mingzhe Li, Di Laura Chen, Mingming Fan, and Khai N. Truong. 2019. FMT: A Wearable Camera-Based Object Tracking Memory Aid for Older Adults. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 3 (Sept. 2019), 95:1–95:25. https://doi.org/10.1145/3351253

[81] Michael Xieyang Liu, Jane Hsieh, Nathan Hahn, Angelina Zhou, Emily Deng, Shaun Burley, Cynthia Taylor, Aniket Kittur, and Brad A. Myers. 2019. Unakite: Scaffolding Developers' Decision-Making Using the Web. In *Proceedings of the 32Nd Annual ACM Symposium on User Interface Software and Technology (UIST '19)*. ACM, New Orleans, LA, USA, 67–80. https://doi.org/10.1145/3332165.3347908 event-place: New Orleans, LA, USA.

[82] Michael Xieyang Liu, Aniket Kittur, and Brad A. Myers. 2021. To Reuse or Not To Reuse? A Framework and System for Evaluating Summarized Knowledge. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW1 (April 2021), 166:1–166:35. https://doi.org/10.1145/3449240

[83] Alan MacCormack, John Rusnak, and Carliss Y. Baldwin. 2006. Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code. *Management Science* 52, 7 (July 2006), 1015–1030. https://doi.org/10.1287/mnsc.1060.0552 Publisher: INFORMS.

[84] Alireza Mansouri, Lilly Suriani Affendey, and Ali Mamat. 2008. Named entity recognition approaches. *International Journal of Computer Science and Network Security* 8, 2 (2008), 339–344. Publisher: Citeseer.

[85] Catherine C. Marshall and Sara Bly. 2005. Saving and Using Encountered Information: Implications for Electronic Periodicals. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '05)*. ACM, New York, NY, USA, 111–120. https://doi.org/10.1145/1054972.1054989 event-place: Portland, Oregon, USA.

[86] Emerson Murphy-Hill, Chris Parnin, and Andrew P. Black. 2012. How We Refactor, and How We Know It. *IEEE Transactions on Software Engineering* 38, 1 (Jan. 2012), 5–18. https://doi.org/10.1109/TSE.2011.41 Conference Name: IEEE Transactions on Software Engineering.

[87] Phong H. Nguyen, Kai Xu, Andy Bardill, Betul Salman, Kate Herd, and B.L. William Wong. 2016. SenseMap: Supporting browser-based online sensemaking through analytic provenance. In *2016 IEEE Conference on Visual Analytics Science and Technology (VAST)*. 91–100. https://doi.org/10.1109/VAST.2016.7883515

[88] Srishti Palani, Zijian Ding, Austin Nguyen, Andrew Chuang, Stephen MacNeil, and Steven P. Dow. 2021. CoNotate: Suggesting Queries Based on Notes Promotes Knowledge Discovery. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3411764.3445618

[89] Alexandra Papoutsaki, James Laskey, and Jeff Huang. 2017. SearchGazer: Webcam Eye Tracking for Remote Studies of Web Search. In *Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval (CHIIR '17)*. Association for Computing Machinery, New York, NY, USA, 17–26. https://doi.org/10.1145/3020165.3020170

[90] Alexandra Papoutsaki, Patsorn Sangkloy, James Laskey, Nediyana Daskalova, Jeff Huang, and James Hays. 2016. WebGazer: Scalable Webcam Eye Tracking Using User Interactions. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*. AAAI, 3839–3845.

[91] Soya Park, April Yi Wang, Ban Kawas, Q. Vera Liao, David Piorkowski, and Marina Danilevsky. 2021. Facilitating Knowledge Sharing from Domain Experts to Data Scientists for Building NLP Models. In *26th International Conference on Intelligent User Interfaces (IUI '21)*. Association for Computing Machinery, New York, NY, USA, 585–596. https://doi.org/10.1145/3397481.3450637

[92] Priyadarshini Patil, Prashant Narayankar, Narayan D.G., and Meena S.M. 2016. A Comprehensive Evaluation of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowfish. *Procedia Computer Science* 78 (Jan. 2016), 617–624. https://doi.org/10.1016/j.procs.2016.02.108

[93] Sharoda A. Paul and Meredith Ringel Morris. 2009. CoSense: Enhancing Sensemaking for Collaborative Web Search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 1771–1780. https://doi.org/10.1145/1518701.1518974

[94] Ksenia Peguero, Nan Zhang, and Xiuzhen Cheng. 2018. An Empirical Study of the Framework Impact on the Security of JavaScript Web Applications. In *Companion Proceedings of the The Web Conference 2018 (WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 753–758. https://doi.org/10.1145/3184558.3188736 event-place: Lyon, France.

[95] Pinterest. [n. d.]. Pinterest. https://www.pinterest.com/

[96] Peter Pirolli and Stuart Card. 2005. The Sensemaking Process and Leverage Points for Analyst Technology as Identified Through Cognitive Task Analysis. In *Proceedings of International Conference on Intelligence Analysis*. http://www.phibetaiota.net/wp-content/uploads/2014/12/Sensemaking-Process-Pirolli-and-Card.pdf

[97] Luca Ponzanelli, Alberto Bacchelli, and Michele Lanza. 2013. Seahawk: Stack Overflow in the IDE. In *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, San Francisco, CA, USA, 1295–1298. https://doi.org/10.1109/ICSE.2013.6606701

[98] Soujanya Poria, Erik Cambria, Lun-Wei Ku, Chen Gui, and Alexander Gelbukh. 2014. A rule-based approach to aspect extraction from product reviews. In *Proceedings of the second workshop on natural language processing for social media (SocialNLP)*. 28–37.

[99] Napol Rachatasumrit, Gonzalo Ramos, Jina Suh, Rachel Ng, and Christopher Meek. 2021. ForSense: Accelerating Online Research Through Sensemaking Integration and Machine Research Support. In *26th International Conference on Intelligent User Interfaces (IUI '21)*. Association for Computing Machinery, New York, NY, USA, 608–618. https://doi.org/10.1145/3397481.3450649

[100] Paruj Ratanaworabhan, Benjamin Livshits, and Benjamin G. Zorn. 2010. JSMeter: Comparing the Behavior of JavaScript Benchmarks with Real Web Applications. In *Proceedings of the 2010 USENIX Conference on Web Application Development (WebApps'10)*. USENIX Association, Berkeley, CA, USA, 3–3. http://dl.acm.org/citation.cfm?id=1863166.1863169 event-place: Boston, MA.

[101] John Rieman. 1993. The diary study: a workplace-oriented research tool to guide laboratory efforts. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems (CHI '93)*. Association for Computing Machinery, New York, NY, USA, 321–326. https://doi.org/10.1145/169059.169255

[102] Kerry Rodden and Xin Fu. 2007. Exploring how mouse movements relate to eye movements on web search results pages. (2007).

[103] Kerry Rodden, Xin Fu, Anne Aula, and Ian Spiro. 2008. Eye-mouse coordination patterns on web search results pages. In *CHI '08 Extended Abstracts on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 2997–3002. https://doi.org/10.1145/1358628.1358797

[104] Tobias Roehm, Rebecca Tiarks, Rainer Koschke, and Walid Maalej. 2012. How Do Professional Developers Comprehend Software?. In *Proceedings of the 34th International Conference on Software Engineering (ICSE '12)*. IEEE Press, Piscataway, NJ, USA, 255–265. http://dl.acm.org/citation.cfm?id=2337223.2337254

[105] Nirmal Roy, Manuel Valle Torre, Ujwal Gadiraju, David Maxwell, and Claudia Hauff. 2021. Note the Highlight: Incorporating Active Reading Tools in a Search as Learning Environment. In *Proceedings of the 2021 Conference on Human Information Interaction and Retrieval (CHIIR '21)*. Association for Computing Machinery, New York, NY, USA, 229–238. https://doi.org/10.1145/3406522.3446025

[106] Daniel M. Russell, Mark J. Stefik, Peter Pirolli, and Stuart K. Card. 1993. The Cost Structure of Sensemaking. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems (CHI '93)*. ACM, New York, NY, USA, 269–276. https://doi.org/10.1145/169059.169209

[107] N. Rutar, C. B. Almazan, and J. S. Foster. 2004. A comparison of bug finding tools for Java. In *15th International Symposium on Software Reliability Engineering*. 245–256. https://doi.org/10.1109/ISSRE.2004.1

[108] Caitlin Sadowski, Kathryn T. Stolee, and Sebastian Elbaum. 2015. How Developers Search for Code: A Case Study. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2015)*. ACM, New York, NY, USA, 191–201. https://doi.org/10.1145/2786805.2786855

[109] Bill N. Schilit, Gene Golovchinsky, and Morgan N. Price. 1998. Beyond paper: supporting active reading with free form digital ink annotations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '98)*. ACM Press/Addison-Wesley Publishing Co., USA, 249–256. https://doi.org/10.1145/274644.274680

[110] M. C. schraefel, Yuxiang Zhu, David Modjeska, Daniel Wigdor, and Shengdong Zhao. 2002. Hunter Gatherer: Interaction Support for the Creation and Management of Within-web-page Collections. In *Proceedings of the 11th International Conference on World Wide Web (WWW '02)*. ACM, New York, NY, USA, 172–181. https://doi.org/10.1145/511446.511469

[111] Ben Shneiderman. 2020. Human-Centered Artificial Intelligence: Reliable, Safe & Trustworthy. *International Journal of Human–Computer Interaction* 36, 6 (April 2020), 495–504. https://doi.org/10.1080/10447318.2020.1741118 Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/10447318.2020.1741118.

[112] Jonathan Sillito, Gail C. Murphy, and Kris De Volder. 2006. Questions Programmers Ask During Software Evolution Tasks. In *Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering (SIGSOFT '06/FSE-14)*. ACM, New York, NY, USA, 23–34. https://doi.org/10.1145/1181775.1181779

[113] Amit Singhal and others. 2001. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.* 24, 4 (2001), 35–43.

[114] Jean Y. Song, Stephan J. Lemmer, Michael Xieyang Liu, Shiyan Yan, Juho Kim, Jason J. Corso, and Walter S. Lasecki. 2019. Popup: reconstructing 3D video using particle filtering to aggregate crowd responses. In *Proceedings of the 24th International Conference on Intelligent User Interfaces (IUI '19)*. Association for Computing Machinery, Marina del Ray, California, 558–569. https://doi.org/10.1145/3301275.3302305

[115] J Stylos and Brad A. Myers. 2006. Mica: A Web-Search Tool for Finding API Components and Examples. In *Visual Languages and Human-Centric Computing (VL/HCC'06)*. 195–202.

[116] Jeffrey Stylos, Brad A. Myers, and Andrew Faulring. 2004. Citrine: providing intelligent copy-and-paste. In *Proceedings of the 17th annual ACM symposium on User interface software and technology (UIST '04)*. Association for Computing Machinery, New York, NY, USA, 185–188. https://doi.org/10.1145/1029632.1029665

[117] Wei Sun, Franklin Mingzhe Li, Benjamin Steeper, Songlin Xu, Feng Tian, and Cheng Zhang. 2021. TeethTap: Recognizing Discrete Teeth Gestures Using Motion and Acoustic Sensing on an Earpiece. In *26th International Conference on Intelligent User Interfaces (IUI '21)*. Association for Computing Machinery, New York, NY, USA, 161–169. https://doi.org/10.1145/3397481.3450645

[118] Craig S. Tashman and W. Keith Edwards. 2011. Active reading and its discontents: the situations, problems and ideas of readers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. Association for Computing Machinery, New York, NY, USA, 2927–2936. https://doi.org/10.1145/1978942.1979376

[119] Simon Tong and Jeff Dean. 2008. United States Patent: 7350187 - System and methods for automatically creating lists. https://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&u=%2Fnetahtml%2FPTO%2Fsearch-adv.htm&r=1&p=1&f=G&l=50&d=PTXT&S1=7,350,187.PN.&OS=pn/7,350,187&RS=PN/7,350,187

[120] Michael L Van De Vanter. 2002. The documentary structure of source code. *Information and Software Technology* 44, 13 (Oct. 2002), 767–782.

[121] Laton Vermette, Shruti Dembla, April Y. Wang, Joanna McGrenere, and Parmit K. Chilana. 2017. Social CheatSheet: An Interactive Community-Curated Information Overlay for Web Applications. *Proc. ACM Hum.-Comput. Interact.* 1, CSCW (Dec. 2017), 102:1–102:19. https://doi.org/10.1145/3134737

[122] Austin R. Ward and Robert Capra. 2021. OrgBox: Supporting Cognitive and Metacognitive Activities During Exploratory Search. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 2570–2574. https://doi.org/10.1145/3404835.3462790

[123] Ryen W. White, Bill Kules, Steven M. Drucker, and m c schraefel. 2006. Supporting Exploratory Search, Introduction, Special Issue, Communications of the ACM. *Commun. ACM* 49 (April 2006), 36–39. https://eprints.soton.ac.uk/263649/

[124] Sungjoon Steve Won, Jing Jin, and Jason I. Hong. 2009. Contextual web history: using visual and contextual cues to improve web browser history. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1457–1466. https://doi.org/10.1145/1518701.1518922

[125] Amy X. Zhang and Justin Cranshaw. 2018. Making Sense of Group Chat Through Collaborative Tagging and Summarization. *Proc. ACM Hum.-Comput. Interact.* 2, CSCW (Nov. 2018), 196:1–196:27. https://doi.org/10.1145/3274465