CheatSheet: A Contextual Interactive Memory Aid for Web Applications

Laton Vermette[*] University of Waterloo
Parmit Chilana[†] University of Waterloo
Michael Terry[‡] University of Waterloo
Adam Fourney[§] University of Waterloo
Ben Lafreniere[¥] University of Saskatchewan
Travis Kerr[**] University of Waterloo

ABSTRACT

We present CheatSheet, a novel contextual interactive memory aid that helps users track their learning progress and refind information when working with complex web applications. Unlike most refinding systems that rely on background monitoring of search sessions or browsing histories to automatically suggest content to users, our approach actively engages users in assessing and curating helpful content for later use. Users create application-specific notes using CheatSheet that contain the visual state of the application overlaid with any text or diagram annotations. Users can also extract snippets of relevant help and tips from other web resources (or other users) and link them to their application-specific CheatSheet. Instead of having to remember or scour through previous notes, bookmarks, or folders, CheatSheet automatically retrieves the recently added notes within the application's user interface. We discuss findings from formative interviews that we used to derive a set of design goals for designing an interactive memory aid, present the design and implementation of CheatSheet, and report on an observational user study that sheds light on the range of users' note-taking and refinding strategies that CheatSheet was able to successfully support.

Keywords: Software learning; software help; memory aid.

Index Terms: H.5.2. Information interfaces and presentation: User Interfaces—Graphical user interfaces.

1. INTRODUCTION

The complexity of modern software applications often requires end users to accomplish a task or solve a problem by trial-and-error or by scouring for examples, tutorials, Q&A, or other forms of help on the Web. However, users can face a key problem by relying on procedural and example-based help resources: while such resources are effective in rapidly improving a user's performance in the short-term, the user may fail at demonstrating the same knowledge and skills after a time lapse or in a new context requiring transfer [12]. For example, an office worker may successfully follow a tutorial to configure complex steps in an enterprise application, but later struggle in remembering even the basic steps when attempting to help a colleague with a similar task. Furthermore, the user may not even remember which resource was used in the learning process or how it was helpful.

In this paper, we investigate a novel approach for helping users track their learning progress and refind information during the use of a web application. Unlike other systems for refinding information on the Web that rely on background monitoring of search sessions or web browsing histories (*e.g.,* [11][14][24]) to automatically suggest content, our approach puts the onus on users to reflect on their interaction and curate helpful content for later use. This approach is inspired by the idea that when learners draw, articulate a self-explanation, or annotate, they create an external representation of the current state of their understanding, which not only helps in reinforcing what they learned, but also later serves as a useful memory aid [1][4][7].

We present *CheatSheet*, a contextual interactive memory aid for web applications that helps users create, organize, and retrieve application-specific annotations, notes, and screenshots within the user interface (Figure 1). One of the key findings from our formative study was that users typically go back-and-forth between a variety of learning resources, such as tutorials, videos, and Q&A sites when proactively learning an applicataion. To support this, CheatSheet allows users to capture and annotate snippets of relevant help and tips from any web-based resource to augment their application-specific notes. CheatSheet associates all annotated notes with particular web applications and automatically loads previously created notes in the side bar of the relevant application. CheatSheet also provides facilities for users to organize, browse, search, and easily share their CheatSheet library or individual notes with other users.

The design of CheatSheet evolved from multiple design iterations inspired by a formative study that examined software users' proactive learning strategies and a second, task-based observational study using the initial CheatSheet prototype. The range of memory aids generated by users and the users' overall positive reactions to CheatSheet provide validation that CheatSheet is highly versatile in supporting a wide range of software learning tasks. In our discussion, we tackle some of the broader issues of refinding and retention issues in the context of learning software and the potential design space of both manual and automatic solutions for creating cheat sheets.

This paper makes the following contributions:

- Empirical findings from an interview study and observational studies that highlight the diversity of software users' proactive learning strategies.
- A novel interface for creating a CheatSheet, an application-specific memory aid that can include annotations, notes, and screenshots of the current application and snippets of helpful resources from the Web.
- An interactive retrieval interface that automatically retrieves a user's CheatSheet containing curated helpful content and notes within an application's user interface. Users can further organize, search, and share their CheatSheets with other users. Importantly, our implementation can work with any web application, making it accessible and usable by millions of users using web applications every day.

2. RELATED WORK

While the concept of helping users track their learning progress with a "cheat sheet" has not been explicitly explored in software learning research, we discuss several other tools for learning, information refinding, and web annotation that have elements related to our CheatSheet system.

{[*]lvermett,[†]pchilana,[**]tjkerr}@uwaterloo.ca
{[‡]mterry,[§]afourney}@cs.uwaterloo.ca
{[¥]ben.lafreniere}@usask.ca

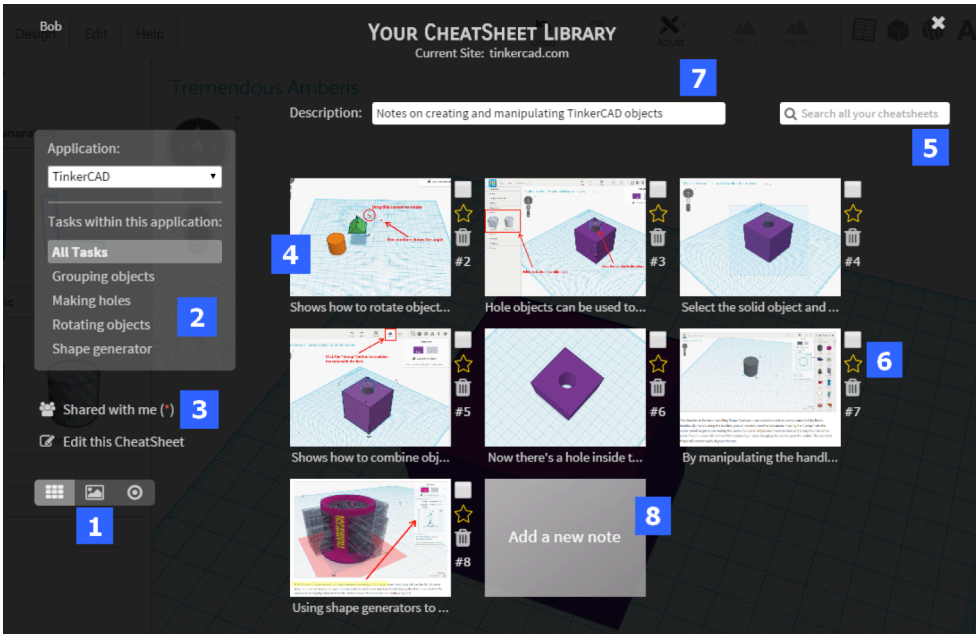

Figure 1: The CheatSheet Gallery interface displays a user's notes related to an application and allows the users to: (1) toggle between three different views (grid, carousel, and radial); (2) filter the notes based on the task; (3) view shared notes or select multiple notes for editing (i.e., deleting, sharing); (4) zoom-in to the note by hovering over a thumbnail and see the related description as a tooltip (not shown); (5) search for notes across the user's library; (6) mark notes as favorites or delete them; (7) view the overall task description; and, (8) add a new note.

2.1. Software Learning Tools

There is a growing body of HCI research concerned with helping users to effectively use complex software applications. Recent examples include multimedia systems that employ videos and animations to demonstrate use [15][22][27], interactive expert-shadowing and tutorial generation techniques [8][13][16][20], and integration of community-generated examples and solutions in the application [6][23]. These types of standalone help systems encourage learning by demonstrating procedures and showing examples, but do not explicitly consider what happens when a user needs to recreate her steps after a time lapse and does not have access to the same resource. Also, as indicated by our formative study, users rarely use a single learning resource when trying to learn a new application—they may begin with a tutorial and find that only one step is useful and then switch to another tutorial or improvise some steps along the way, or search for and incorporate additional examples from the Web. With CheatSheet, the user can manually curate the content and decide which tutorial steps, answers, or video frames in the help-seeking process were actually useful. Instead of trying to remember where these notes were saved, the user can automatically see all application-specific notes in the side bar of the application's user interface.

2.2 Tools for Refinding and Organizing Web Content

The problem of refinding information on the Web [5] has inspired numerous systems in information retrieval and HCI to help users locate previously seen content. For example, refinding systems have been developed for desktop use [11], search queries, [24], and web browsing sessions [30]. However, CheatSheet only automates the process of retrieving users' application-specific notes and provides full manual control in selecting and curating content that users actually find useful or anticipate using later.

Some systems that have elements related to CheatSheet include tools that allow users to select and summarize content from the Web (*e.g.,* [9][10]). But, unlike CheatSheet, these systems are driven by the goal of generating templates and building automated ways of collecting similar web content, rather than creating a memory aid.

Systems that explicitly support refinding when learning a software application are only beginning to emerge. For example, *InterTwine* [14] provides an automatic linkage between the Web browser and a desktop application to generate a shared interapplication history and assist with task-specific refinding. However, the underlying assumption here is that users mainly access help through a search engine. In contrast, CheatSheet does not make any such assumption and tries to facilitate a wide range of users' needs in assessing, selecting, and curating help content from the Web.

2.3 Tools for Recording Notes and Annotations

The annotation aspect of CheatSheet builds off of a rich history of document annotation and note-taking systems, even though they have not been designed for the purpose of learning software. For example, *Margin Notes* [28] automatically annotates and rewrites a web page by monitoring a user's past actions on the page. On the other hand, systems such as *MADCOW* [3] and *SpreadCrumbs* [19] allow users to insert annotations within specific locations in a web page. Although relevant, a limitation of these annotation approaches is that they tend to be obtrusive and distracting because they change the underlying HTML DOM and appear in-line every time a user visits the page. In contrast, users can turn CheatSheet on or off at any point, and can also open their application-specific CheatSheet in another tab or window.

CheatSheet's screen annotation features are most closely related to *ScreenCrayons* [25] where users can add notes and highlights to any captured screen. However, our technical contribution is not in being able to annotate screens, but rather in being able to create application-specific notes and linkages for learning a new application. For example, our system can support CheatSheet creation for a task sequence by automatically collecting relevant screens based on the user's input and traversal across a variety of resources (tutorials, Q&A sites, etc.).

And, finally, there are commercial tools, such as *Evernote* and *OneNote* that support general note-taking on the Web and offer several clipping and annotation features. CheatSheet differs from these systems in significant ways: 1) CheatSheet uses the visual representation of the page as the default platform for further annotation and linking to facilitate recall; 2) CheatSheet allows for a template of notes to be generated semi-automatically for a task sequence by monitoring users' input activities; and 3) CheatSheet automatically retrieves users' notes relevant to the current application and offers facilities for users to rearrange and select notes they want to see when the application is re-accessed (functioning more like a physical cheat sheet for the application, rather than a generic notebook).

In summary, while CheatSheet is inspired by a wide range of software learning, information refinding, and document annotation tools, our design is among the first to function as a lower-friction memory aid that helps software learners track their progress and assess, curate, organize, and retrieve application-specific help content.

3. Proactive Software Learning practices: A Formative Study

To inform the design of CheatSheet, we first carried out formative interviews with 12 adult software users who were proactively learning a software application by enrolling in a course or by using web resources to learn on their own.

3.1 Participants and Procedure

Our 12 interviewees (6 males, 6 females) were between the ages of 19-64 and included office workers, scientists, professors, and undergraduate and graduate students. These interviewees described a range of complex applications that they were trying to learn for work or school-related tasks, including statistical analysis tools, office software, citation management systems, web development tools, and scientific data analysis tools. (Only one interviewee had any formal training in Computer Science and the majority described themselves as "non-technical" users). We carried out semi-structured interviews that lasted between 30-45 minutes and queried about users' strategies for re-accessing or re-using previously learned functionality. We also inquired about the extent to which they proactively took notes, used annotations, or wrote self-explanations. Finally, we asked the interviewees to fill out a brief demographic questionnaire. We transcribed and analyzed the data using an inductive analysis approach to distill recurring themes in interview responses.

3.2 Key Findings and Design Goals

Over two-thirds of our interviewees explained that they always used some form of note taking when learning a new application, because they were not confident about remembering all the details and nuances upon first exposure. The remaining interviewees said that they initially used no particular strategy, but often became more proactive about taking notes when they struggled to re-find a previously learned functionality. All interviewees also described social uses of their notes, whereby they either asked for help or shared snippets of helpful information with others.

Overall, we found that learners placed the most emphasis on retaining *visual cues* related to the current task or workspace: *"you just need something to jog your memory…they call it the 'light bulb effect'" (P09)*. Participants sometimes created print-outs of screens and annotated them by hand to retain such cues. Other users preferred electronic versions for taking notes—for example, they would take screenshots of their current window and paste them into tools, such as *MS Word* and *MS PowerPoint*: *"I take notes and I also take pictures, screenshots, so later when I am doing the other stuff, having the application [view] there is nice… I usually send an email with notes to myself…" (P10)*.

While having the visuals was important, more than half of the interviewees also valued textual explanations of the needed steps. A few of these interviewees particularly emphasized having explanations *"in their own language."* Other interviewees explained how they wrote notes to remind themselves of any unexpected nuances: *"When I find that something works in a surprising way, I always write that down. I have a separate section for that, just like little reminders for myself to say this doesn't work the way you think it does" (P12)*.

One of the key findings from our study was that learning software involved a complex interplay between a variety of different resources: switching between the application, a tutorial, a help video, documentation, a web forum, or asking another user or expert. More than half our interviewees described strategies involving annotations of links or snippets taken from web resources, and efforts to store them in a meaningful way, such as in annotated emails, text files, and bookmarks: *"…if I am learning something new, like a programming language, I will make a little cheat sheet for myself, so I will have a reference file…really just a bullet list of commands I use frequently and then a short explanation of what to do if it is not clear...and maybe URLs" (P05)*.

One challenge identified by the interviewees was being able to organize and manage application-specific notes. Interviewees described how it was useful to initially have their notes saved on their desktop or in folders that could be easily accessed, but over time, these notes became more scattered and difficult to locate. Participants mentioned saving notes in email or text files so that they could use built-in search features, but they often had difficulty in coming up with a useful query to look up what they had actually saved. Some interviewees mentioned they would use browsers' bookmark features, but ended up having *too many* bookmarks over time.

Based on findings from this formative study and the known benefits [1][4][7] of actively engaging users in learning and information finding tasks, we synthesized four main design goals for CheatSheet:

1 **Exploit visual memory channels:** Since software users find visuals to be the most effective when trying to remember complex software features and functions, they should have access to a lightweight method for generating visual forms of notes.
2 **Facilitate aggregation of help resources:** Since the use of an application and help resources is often intertwined with the learning process, users should be able to easily aggregate, annotate, organize, and locate "snippets" of information from a variety of resources, in addition to application-specific visuals.
3 **Allow for in-application contextual retrieval**: Users should be able to quickly retrieve and recognize their notes in the context of the application they are currently using.
4 **Support social sharing and learning:** Users should have quick access for sharing their notes with other users and collecting helpful snippets shared by other users.

4. CheatSheet User Interface: The Initial Design

Inspired by our design goals, we designed and implemented CheatSheet as a browser plug-in that can work with any web application. Our initial design allowed users to access CheatSheet with a single click on the browser's menu toolbar. The main components of the CheatSheet system are the *CheatSheet Gallery* (Figure 1) and the *CheatSheet Canvas* (Figure 2). Users create an account for their first use and remain logged in to the CheatSheet system throughout their browser session.

To describe the main user interface and functionality of CheatSheet, we begin with an example usage scenario.

4.1 Example Usage Scenario

Bob is learning about 3D printing for the first time and is trying out *Tinkercad*, an online computer-aided design (CAD) application to model his 3D object. He has a Tinkercad file open and is following an online tutorial on creating a birdhouse. He first inserts a cylinder and a roof-shaped object and tries to follow some complex instructions on rotating and merging both of the objects. Even though Bob followed all the steps, he does not feel confident that he understands it well enough to do it on his own next time and decides to create a CheatSheet by accessing his recently installed browser plugin.

Bob calls his CheatSheet "Tinkercad" and adds a new note. The CheatSheet system automatically captures an image of the application window in a new canvas (Figure 2) and displays a variety of annotation tools (Figure 2.1) and metadata fields for description (Figure 2.2-2.7).

Bob first uses the ellipse tool to draw attention to the rotation feature, and then adds arrows and text to describe the steps. He also adds a description in the text field (Figure 2.3) to remember what he actually did in his own words, and specifies the task (Figure 2.4) to be "rotating objects." When he presses save, he is taken to the gallery view (Figure 1) where he can browse and organize his other notes for his Tinkercad CheatSheet.

Bob goes back to the tutorial and notices that there were some useful descriptions in the tutorial for key concepts such as *"workplane"* that he wants to remember for later. On the tutorial site, he accesses his CheatSheet library, and now selects the existing Tinkercad CheatSheet to add his note. He crops out a snippet of the particular area of the tutorial page that he wants to save and tags it as "definitions." Now, he sees that both his annotated image and tutorial snippet appear as part of the Tinkercad CheatSheet.

Later that day, Bob's colleague Chris emails him a question about rotating objects in Tinkercad. Bob visits the Tinkercad application and opens up his CheatSheet, where he is automatically shown his Tinkercad notes. He finds his note on rotating objects and since Chris is already a user of CheatSheet, Bob is able to find him quickly by typing his name (Figure 2.5) and share the note. The note immediately gets added to Chris' CheatSheet Library.

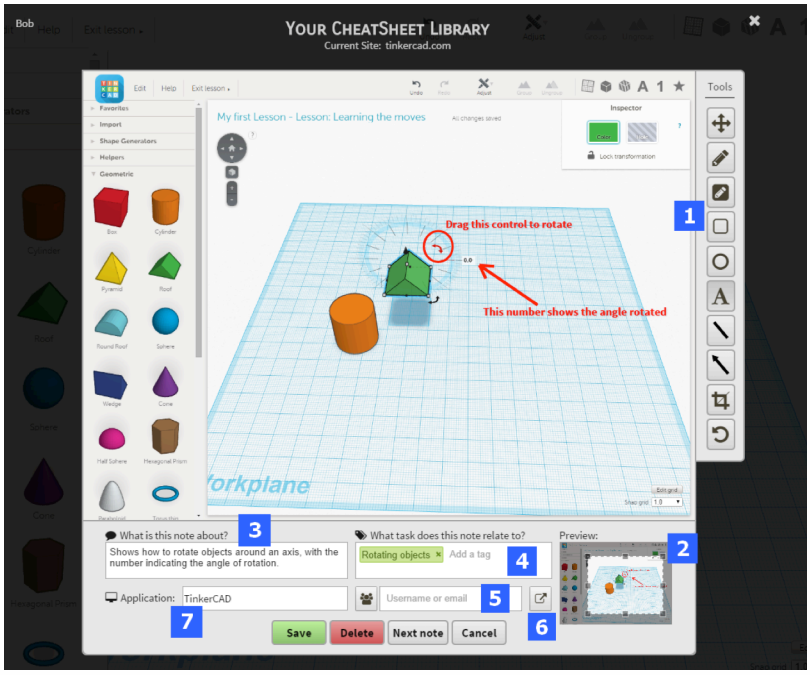

Figure 2: The CheatSheet canvas displays a screen capture of the current view and contains: (1) a toolbar with various annotation tools; (2) a preview window to set the thumbnail view for the note; (3) a textbox for description; (4) tags to describe the task; (5) a feature to share the note by specifying a username or email; (6) a bookmark to the page where the screenshot was captured; (7) a tag to specify the application name.

4.2 CheatSheet Canvas for Creating Notes

Users add new notes to their CheatSheet library by clicking on the "add a note" link (Figure 1.6) from the Gallery view and edit and save content through the CheatSheet Canvas (Figure 2).

4.2.1 Visual Capture of Context and Annotation

To support our design goal of exploiting visual memory channels, CheatSheet automatically captures and places a screenshot of the user's current window centered on the canvas when a user adds a note. On the right side of the canvas, various annotation tools are available to mark up the screenshot (Figure 2.1), such as highlights, shapes, arrows, and text. Users can also crop a specific portion of the image.

Users can enter their own description or edit text that can be automatically extracted from a web resource by simply highlighting it on the screen before an image is captured. As the user edits the canvas, the preview window (Figure 2.2) shows a real-time preview of the thumbnail that will be generated and shown in the gallery interface. Users can resize and pan through the preview to highlight particular annotations.

To retain the original context, when a new note is added, the system automatically saves the URL of the page, allowing users to have a "bookmark" to the actual page. This link can be accessed from the toolbar (Figure 2.6).

4.2.2 Aggregation of Relevant Help Content

To support our design goal of aggregating and organizing notes from disparate web resources, CheatSheet allows users to "tag" their note (Figure 2.4) with the name of an application and any specific task that the note is relevant to (the default tag is "general"). The system will autosuggest tags that have already been used in the CheatSheet library and these tags can be further used in the gallery interface (Figure 1.2) to filter notes.

4.2.3 Social Sharing and Collaboration

To support our design goal of social sharing, the system provides a simple mechanism (Figure 2.5) to quickly share a note with another user who is already using the CheatSheet system through a simple lookup of usernames (common in social sharing sites). Recipients of these notes can annotate and resend these notes back to the senders (or other users) within the CheatSheet environment.

4.3 CheatSheet Gallery for Browsing Notes

In the Gallery view (Figure 1), CheatSheet overlays thumbnails of a user's notes for the current application. (The current application

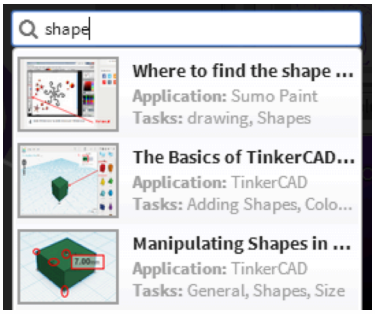

Figure 3: CheatSheet's autocomplete search feature

is detected by matching the domain name of the application's URL). Users have several options to browse, update, and organize their existing library of notes.

4.3.1 Browsing Notes in the CheatSheet

To support our design goal of facilitating in-application contextual retrieval and to promote long-term use of notes, annotations, and screenshots, CheatSheet offers several browsing and searching features.

CheatSheet initially shows users' existing notes as thumbnails sorted by creation time, in a grid view (Figure 1). (The thumbnails are laid out according to the preview set by the user in the Canvas, as shown in Figure 2.2). Users can use the task-based browsing feature to filter notes related to different tasks within the application. For example, in the usage scenario, Bob may have several notes related to the Tinkercad application, but can quickly filter to find notes related to the rotate function (Figure 1.2). Users can also rearrange the notes by simply selecting and dragging them in the gallery.

If the user has received shared notes from other users of the CheatSheet system, she can view them by clicking on "Shared With Me" (Figure 1.3).

4.3.2 Thumbnails and Zoom-in Interaction

In the gallery layout, users can hover over thumbnails of notes to quickly zoom into a certain location or to see the description of the note as a tooltip (clicking on the note opens it in the canvas). We adapted this idea from similar designs, such as *Data Mountain* [29], where the use of thumbnails and tooltips leverages the user's spatial and visual memory and helps them remember previously viewed content.

4.3.3 Searching For Notes

As users' CheatSheet libraries grow over time with multiple tasks across different applications, users can rely on autocomplete keyword search (Figure 3) to quickly find notes based on task name, application name or the associated description. Search results display thumbnails of matching results and the associated metadata.

4.4 Implementation

We implemented CheatSheet as a Chrome browser extension, although the core functionality can be easily adapted to other browsers. CheatSheet is independent of any knowledge of the implementation of the underlying web application. All notes and user-related information in the CheatSheet system are currently stored in a database on our third party server. The history of a user's interactions in CheatSheet is stored and retrieved from the client side. When a user adds a new note, the system automatically uses a plug-in to capture the currently visible tab as an image and captures any selected text to populate the description field.

5. Formative User Evaluation with CheatSheet

To understand how users would perceive the initial design of CheatSheet and what type of note-taking strategies they would exhibit using this medium, we carried out an observational scenario-based user study.

5.1 Participants and Procedure

We recruited 13 participants for this user study (7 females, 6 males). They were between the ages of 18-49 and had educational backgrounds ranging from high school diplomas to graduate degrees. All but four of the participants spoke English as a first language and only two of the participants had any formal training in Computer Science.

The user evaluation was carried out in a single session lasting 45-60 minutes. We began each session with a brief three-minute overview of the CheatSheet system and showed participants how to interact with various CheatSheet features using *Google Maps* as an example application.

Users were asked to complete two tasks (described below) using CheatSheet. Next, participants filled out a brief post-task questionnaire that collected Likert-scale responses on a scale of 1-7 about their perceptions of the system and demographic information. Lastly, we conducted follow-up interviews to probe into participants' use of different CheatSheet features and their overall reactions. We performed the study using our custom Chrome (v33.0) extension on a Windows 7 laptop machine with 12GB RAM and 2.3GHz processor, connected to a 17-inch monitor. We captured participants' mouse movements and on-screen activity using screen capture software, and audio-recorded and transcribed users' answers to interview questions.

5.2 Study Tasks

For our first task, we told participants that they were trying to learn a new online photo editing application called *Pixlr*. With two images open, their task was to learn how they could move a boat from the first image to the water in the second image. We gave users the option to use their own knowledge to perform this task or to use an online tutorial (open in another tab) that explained how to "cut" and paste objects from images using the polygonal lasso tool. In doing this task, we asked the participants to imagine that they will need to perform this task again, and to use the CheatSheet system to make notes for later access.

For the second task, we told participants that their boss had data on sales by each employee over a one-year period (provided as a *Google spreadsheet*). He was interested in calculating averages, but was not familiar with spreadsheet functions. We asked participants to create a CheatSheet for showing how to calculate the overall *average sale* for each of the top employees using a pivot table. Participants could use whatever resource they wanted to find help on creating pivot tables if they were not sure.

5.3 Main Findings and Design Implications

Overall, we found that the majority of users quickly became comfortable with CheatSheet upon their first use. Our questionnaire analysis indicated that 85% (11/13) of the participants either strongly agreed, agreed, or somewhat agreed that CheatSheet was easy to use, that they enjoyed using the system, and that they would recommend the system to others.

5.3.1 Using CheatSheet in Proactive Learning

All of our 13 participants consistently appreciated the idea of being able to take screenshots and annotate them so that they could visually remember the steps or retain cues. But, similar to

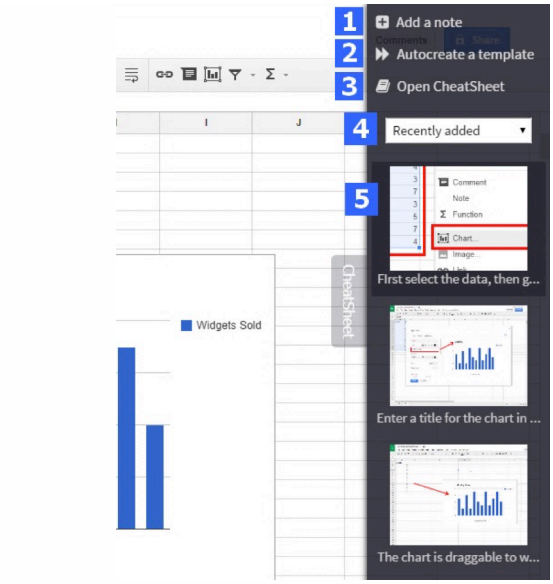

Figure 4: CheatSheet's sidebar implementation that appears automatically when a web application is loaded and allows users to: (1) add a single note; (2) autocreate a template of multiple notes; (3) open the full notes library; (4) toggle between different tags to see all application-specific notes; (5) zoom into note previews.

our formative interviews, and research on different learning and encoding channels [26], we observed that our participants had different preferences for using text and annotations: "*I am a visual learner so I like the pictures...but, at the same time, just because something is circled...like step 1-2-3, this wouldn't help me in the order. The description would help me remember" (U05)*

We also found that 4 participants wrote the textual description "in their own words" to facilitate recall rather than only rely on a screenshot or steps copied from another source. The use of self-explanations has been shown to be one of the most effective externalization strategies [7] and CheatSheet offers the potential to further exploit this strategy by explicitly prompting users to add such explanations and better track their learning progress.

One surprising finding in our study was that another 4 of the participants made notes *only* using snippets from tutorials and other resources; these participants did not use the application screenshots as part of their notes. Some participants extracted snippets based on what they anticipated they might need later in the task; some wanted to note things that did work and did not work; others extracted snippets to note something interesting to try later, even if it was not relevant to the current task: "*I like to combine insights from different web pages...CheatSheet [is] great because not only can I record what I want, but I can also make notes on what I did and how I deviated from the process." (U13)*

As we observed users' different note-taking and learning strategies and probed them about their choices, we found two limitations with CheatSheet's current design: 1) more than half of the users felt that they would sometimes want notes to reflect a task sequence, but it would be too cumbersome to add a single note for each step; and 2) some users wanted a quick keyboard shortcut to signal the capture of an important screen, but did not want to curate it immediately as it interrupted their workflow.

5.3.2 Accessing and Retrieving Notes Through CheatSheet

More than half of our participants appreciated the ease with which they could bring up their previous notes in an application without having to remember where their notes were saved. The main benefit here was that users did not have to rely on keyword search to remember how they had previously located a useful piece of information: *"It [Google search] doesn't always work. You find something different or you find something contradictory or the thing that you thought you found last time, you can't find again" (U03)*

Some participants also commented on the usefulness of the task-specific organizational features offered by CheatSheet, especially in comparison to some of their other more cumbersome refinding strategies, such as opening up multiple tabs in the browser: *"Sometimes I use a session saver and save my tabs so I can go back, but usually by the time I have so many sessions I have no idea which session I have in which tab. So, this [CheatSheet] would be really useful, since it is sorted by application…it can just know that [this note is] specific for this application…" (U11)*

Although all participants found CheatSheet's application-specific retrieval to be an asset, one drawback pointed out by participants was that they may forget to explicitly invoke the plugin through the browser toolbar.

5.3.3 CheatSheet as a Tool for Helping Others

All of our 13 participants were very positive about the social sharing feature and some wanted to begin using CheatSheet immediately, if only to use this feature. Surprisingly, some participants even saw the quick visual sharing feature as being useful in more formal contexts, such as help desks: *"I think it is great for help desk people…like if the user is having a problem configuring Outlook or something…we could give her this link [from CheatSheet]" (U08).* This participant went on to explain that having a lightweight in-browser shared visual context could save a lot of back and forth that usually occurs with troubleshooting: "*…terminology, like what to call this, do you call it start bar, menu bar, file menu bar, those kinds of things. You know like older people or people not technically sound wouldn't exactly know what you are talking about with that terminology."(U08)*

The only limitation of the sharing feature that users noted was that CheatSheet only allowed users to share notes with people who had downloaded the CheatSheet plugin. There was concern that some of the users (who were more likely to need help) would not be able to install a plug-in on their own, making the social sharing feature inaccessible for them (this was addressed in the second design iteration).

6. Second Design Iteration of CheatSheet

Although users were overall positive about the concept of having a system for tracking their learning progress, we found a few critical limitations with CheatSheet's current design. In our second design iteration, we focused on improving CheatSheet in 4 areas to better support users' diverse range of learning needs.

6.1 Providing automatic retrieval of existing notes

To save users the step of clicking on the plugin shortcut in the browser toolbar, we investigated a new design through a sidebar interface (Figure 4) that automatically retrieves a user's notes related to the current domain of the site. By default, CheatSheet displays the notes most recently created (Figure 4.5), but users can also choose to switch the view to see notes that they marked as "favorites" or notes that they created related to other specific tasks. Hovering over any of these notes brings up the full description in a tooltip, and zooms into image preview based on the user's cursor location, giving them a way to quickly review the most relevant notes without leaving the current page.

The sidebar also offers quick access to add a new single note (Figure 4.1) or add multiple notes corresponding to a task with an auto-generated template (Figure 4.2). This second option was designed for users who prefer to focus on their task and take notes afterwards, or who prefer to capture a sequence of steps together (discussed next).

6.2 Generating semi-automatic templates of multiple notes

To facilitate the capture of a sequence of steps, we designed CheatSheet's autocreation process that can be triggered through a start and stop button in the sidebar and browser toolbar. We explored two modes of automatic capture: one which captures notes at regular time intervals (8 sec), and one which captures notes triggered by user interactions (i.e. clicking, scrolling, switching tabs, pressing enter), inspired by systems that allow for macro recording on the Web (*e.g.*, [18][21]). In either case, the system uses duplicate image detection algorithms to check if a captured note is identical to the previous one, and discards these duplicates to minimize the curation work for users. Each captured step is stored as a note and users can review and edit them in the gallery interface (Figure 5). Note that the goal here was only to facilitate the capture of the users' sequence of screens rather than automate the CheatSheet creation process.

6.3 Providing facilities for power users

CheatSheet's new design also provides keyboard shortcuts to support users who prefer more automated access to note creation. One shortcut allows the user to quickly open up the canvas interface (Figure 2) and add a note for editing. The other shortcut allows users to only indicate that a screen is relevant and a note is created in the background, without interrupting the user's workflow. The user can later access this note through the sidebar interface or the main CheatSheet gallery. This might be analogous to the concept of inserting post-it notes while reading a book, but not annotating them until after finishing the book.

6.4 Sharing notes and libraries with other web users

To support sharing with people who may not be CheatSheet users, we added a facility to insert an email address in the sharing field (Figure 2.5) rather than a username. The recipient will automatically be sent an email containing a persistent link to an html page containing the sender's annotated note. Several steps can also be shared at once (i.e., as a tutorial) by accessing the multiple edit feature (Figure 1.3).

7. Second User Evaluation

To obtain feedback for our second design iteration, we recruited another 7 participants (2 females) and asked them to complete the same two tasks using Pixlr and Google SpreadSheets (explained above). We were mainly interested in understanding participants' use of our autocreate feature, the new in-application retrieval, and general feedback.

We found that users were able to quickly grasp the autocreate feature and started using it in the study tasks. We found that users' preference for capture based on time interval versus input events varied depending on context. For example, some users preferred the time-interval based recording to take notes for their own retention: "*I think when I'm doing things for myself, I would prefer the auto feature to take a bunch of notes and then I can just keep the stuff that I want…the 8 second threshold would be enough for me because for I don't need every step…I can look at my notes and then just remember…" (U20).*

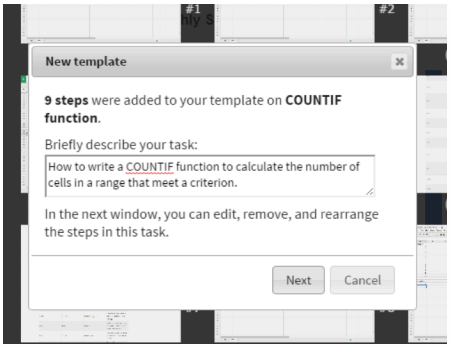

Figure 5: CheatSheet's template feature for capturing and editing multiple notes in a task sequence.

Another user explained that the time-interval approach may miss out on minute steps when explaining things to another user, especially where every mouse click may be significant in a task sequence: "*I often skip stuff when I am explaining something to someone else; for example, I use keyboard shortcuts in Photoshop, or might do 3 or 4 things in one step, which makes it really hard to explain it to other people; this is a huge advantage of auto-capture because I can't skip so far ahead…it'll help me with my pace…and I really like being able to do this right in place [in the user interface]" (U17).*

Other users preferred not to use the autocreate feature at all because it would generate too many steps for them to curate afterwards and liked having control over what they could record. Two of our participants mentioned that they would prefer to use the keyboard shortcuts to manually choose when to capture useful screens in the background.

These insights from users provided validation for our revised design choices to give users more control over choosing whether to record a single screen, or to record multiple screens in a sequence based on input events or time intervals, or to record screens via shortcuts.

In other feedback, participants immediately appreciated the side-bar overlay that retrieved their past notes automatically when they visited the same application again. Being able to choose and rearrange the notes to be displayed in the side bar gave them the feel of creating a real "*cheat sheet*" as it encouraged users to be selective about content they would really need to use.

Finally, the tradeoff of manually capturing and curating notes upfront versus benefits of refinding relevant content later are best summed up by this participant's response: "*I usually don't take notes because it takes time, but I usually waste a lot of time afterwards [to refind]. With this [CheatSheet], it makes me spend more time upfront, but it [will] save me a lot of time later" (U18).*

8. Discussion

As we increasingly rely on software applications for our work and personal activities, fully grasping and retaining knowledge of all the commands, functions, and features in these applications can be a challenge. As we discovered in our formative studies, this problem is particularly acute for the less "tech-savvy" users who are not able to keep up by relying only on their short-term memory or trial-and-error. While it is important for us to invent new procedural and example-driven help resources in HCI, we argue that we also need to tap into other learning approaches that give users an opportunity to better manage and reflect on their learning progress.

Using an iterative user-centred design approach, we have developed CheatSheet, a novel contextual interactive memory aid that helps users create, organize, and retrieve application-specific annotations, notes, and screenshots. It also provides an automatic in-application retrieval interface that allows users to easily refind their existing notes and annotations. We now discuss some of the

implications and limitations of our approach and how they could be addressed in future work.

First, our current user evaluations were mainly used to inform the design of CheatSheet, capture initial user reactions, and shed light on note-taking practices in the context of tracking learning progress. There are a number of research questions about actual long-term retention benefits that require larger controlled studies or field deployments and were beyond the scope of the current paper. There also are other interesting questions about the note-taking mechanisms that can provide different learning benefits, as investigated in other domains: for example, should the interactive memory aid explicitly require users to add a textual explanation [1]? Or, should the system monitor users' activities and nudge them to generate a particular screenshot or annotation [17]? We plan to pursue such research questions in future work (and will also make CheatSheet available to other researchers so that they can investigate other novel questions using our platform.)

Although we focused on designing CheatSheet as a memory aid, a surprising side-effect was the positive response to the lightweight social sharing feature. Users immediately saw the benefit of using it for quickly generating tutorials or for visually asking for help from other users. The use of social features is not only useful for troubleshooting, but can also have potential long-term retention benefits as learners often understand complex concepts best by explaining them to others [2]. In future work, we will extend our designs and prompts to explicitly exploit this learning benefit.

One limitation of CheatSheet's current design and implementation is that it only supports web-based applications. Although the Web is increasingly offering feature-rich complex applications and a number of applications that we tested with CheatSheet have desktop counterparts, many widely used complex applications are still only available on the desktop. In fact, several of our participants requested a desktop version of CheatSheet. We believe that adapting the CheatSheet concept for desktop applications is more of an engineering challenge and that most of the design goals and ideas that we introduced can be easily adapted.

And finally, although we iterated on our designs multiple times to facilitate the note-taking process, the main interaction of CheatSheet still requires explicit manual invocation by the user and interrupts the current task flow. The other extreme may be to explore purely automated approaches [*e.g.*,14] that require little or no user intervention and still offer useful suggestions to users. We believe that there is a large open design space to explore more designs between these extremes and need for more empirical studies to investigate the specific learning benefits. Our overall aim with CheatSheet has been to help users who are proactively learning complex software to be more aware of their learning progress and to improve access to previously helpful web resources.

References

[1] Aleven, V.A. and Koedinger, K.R. 2002. An effective metacognitive strategy: Learning by doing and explaining with a computer-based Cognitive Tutor. *Cognitive science*. 26, 2 (2002), 147–179.

[2] Biswas, G., Leelawong, K., Schwartz, D., Vye, N. and Vanderbilt, T.T.A.G. at 2005. Learning by teaching: A new agent paradigm for educational software. *Applied AI*. 19, 3-4 (2005), 363–392.

[3] Bottoni, P., Civica, R., Levialdi, S., Orso, L., Panizzi, E. and Trinchese, R. 2004. MADCOW: a multimedia digital annotation system. *Proc AVI*, 55–62.

[4] Bransford, J.D., Brown, A.L. and Cocking, R.R. 2000. *How people learn*. National Academy Press Washington, DC.

[5] Bruce, H., Jones, W. and Dumais, S. 2004. Keeping and re-finding information on the Web: What do people do and what do they need? *Proc ASIS&T,* 41, 1 (2004), 129–137.

[6] Chilana, P., Ko, A.J. and Wobbrock, J.O. 2012. LemonAid: selection-based crowdsourced contextual help for web applications. *Proc CHI*, 1549–1558.

[7] Chi, M.T., De Leeuw, N., Chiu, M.-H. and LaVancher, C. 1994. Eliciting self-explanations improves understanding. *Cognitive science*. 18, 3 (1994), 439–477.

[8] Chi, P.-Y., Ahn, S., Ren, A., Dontcheva, M., Li, W. and Hartmann, B. 2012. Mixt: automatic generation of step-by-step mixed media tutorials. *Proc UIST*, 93–102.

[9] Dontcheva, M., Drucker, S.M., Salesin, D. and Cohen, M.F. 2007. Relations, cards, and search templates: user-guided web data integration and layout. *Proc UIST*, 61–70.

[10] Dontcheva, M., Drucker, S.M., Wade, G., Salesin, D. and Cohen, M.F. 2006. Summarizing personal web browsing sessions. *Proc UIST*, 115–124.

[11] Dumais, S., Cutrell, E., Cadiz, J.J., Jancke, G., Sarin, R. and Robbins, D.C. 2003. Stuff I've seen: a system for personal information retrieval and re-use. *Proc SIGIR,* 72–79.

[12] Eiriksdottir, E. and Catrambone, R. 2011. Procedural Instructions, Principles, and Examples How to Structure Instructions for Procedural Tasks to Enhance Performance, Learning, and Transfer. *J Human Factors &Ergonomics Society*. 53, 6 (2011), 749–770.

[13] Fernquist, J., Grossman, T. and Fitzmaurice, G. 2011. Sketch-sketch revolution: an engaging tutorial system for guided sketching and application learning. *Proc UIST*, 373–382.

[14] Fourney, A., Lafreniere, B, Chilana, P and Terry, M 2014. InterTwine: Creating Interapplication Information Scent to Support Coordinated Use of Software. *Proc UIST* (2014).

[15] Grossman, T. and Fitzmaurice, G. 2010. Toolclips: An investigation of contextual video assistance for functionality understanding. *Proc CHI,* 1515–1524.

[16] Grossman, Tovi, Matejka, Justin and Fitzmaurice, George 2010. Chronicle: Capture, Exploration, and Playback of Document Workflow Histories. *Proc UIST*.

[17] Hausmann, R.G. and Chi, M.H. 2002. Can a computer interface support self-explaining. *Cognitive Technology*. 7, 1 (2002), 4–14.

[18] Hupp, D. and Miller, R.C. 2007. Smart bookmarks: automatic retroactive macro recording on the Web. *Proc UIST*, 81–90.

[19] Kawase, R. and Nejdl, W. 2009. A Straightforward Approach for Online Annotations: SpreadCrumbs-Enhancing and Simplifying Online Collaboration. *WEBIST* (2009), 407–410.

[20] Kelleher, C. and Pausch, R. 2005. Stencils-based tutorials: design and evaluation. *Proc CHI*, 541–550.

[21] Li, I., Nichols, J., Lau, T., Drews, C. and Cypher, A. 2010. Here's what i did: sharing and reusing web activity with ActionShot. *Proc CHI*, 723–732.

[22] Matejka, J., Grossman, T. and Fitzmaurice, G. 2011. Ambient help. *Proc CHI*, 2751–2760.

[23] Matejka, J., Grossman, T. and Fitzmaurice, G. 2011. IP-QAT: in-product questions, answers, & tips. *Proc UIST*, 175–184.

[24] Morris, D., Ringel Morris, M. and Venolia, G. 2008. SearchBar: a search-centric web history for task resumption and information re-finding. *Proc CHI*, 1207–1216.

[25] Olsen Jr, D.R., Taufer, T. and Fails, J.A. 2004. ScreenCrayons: annotating anything. *Proc UIST* 165–174.

[26] Paivio, A. 1971. *Imagery and verbal processes*. Holt, Rinehart & Winston.

[27] Pongnumkul, S., Dontcheva, M., Li, W., Wang, J., Bourdev, L., Avidan, S. and Cohen, M.F. 2011. Pause-and-play: automatically linking screencast video tutorials with applications. *Proc UIST*, 135–144.

[28] Rhodes, B.J. 2000. Margin notes: Building a contextually aware associative memory. *Proc IUI*, 219–224.

[29] Robertson, G., Czerwinski, M., Larson, K., Robbins, D.C., Thiel, D. and Van Dantzich, M. 1998. Data mountain: using spatial memory for document management. *Proc UIST*, 153–162.

[30] Won, S.S., Jin, J. and Hong, J.I. 2009. Contextual web history: using visual and contextual cues to improve web browser history. *Proc CHI*, 1457–1466.