Design and Evaluation of Social CheatSheet: A Community-Curated Software Help Overlay

by

Laton Vermette

A thesis

presented to the University of Waterloo

in fulfilment of the

thesis requirement for the degree of

Master of Mathematics

in

Computer Science

Waterloo, Ontario, Canada, 2017

Author's Declaration

Statement of Contributions

This thesis includes ideas and content (including figures and tables) that have been accepted, but not yet published, in peer-reviewed publications. The conference paper from which I have adapted content is the following, which I co-authored:

- **Laton Vermette**, Shruti Dembla, April Wang, Joanna McGrenere, Parmit K. Chilana. 2018. Social CheatSheet: An Interactive Community-Curated Information Overlay for Web Applications. To appear in *Proceedings of the 2018 ACM Conference on Computer Supported Cooperative Work and Social Computing* (CSCW '18). ACM, New York, NY, USA.

Abstract

Software users can often find it difficult to sift through dense help pages, tutorials, Q&A sites, blogs, and other resources, trying to locate useful task-specific instructions for the applications they use. We present Social CheatSheet, an interactive information overlay that can appear atop any existing web application and retrieve relevant step-by-step instructions, tips, and tutorials curated by a community of software users. Based on the findings from two formative studies, the system offers several features for users to search, browse, filter, and bookmark community-generated help content, and to ask questions and seek clarifications. Furthermore, Social CheatSheet includes embedded curation features for users to generate, annotate, and categorize their own visual notes and tutorials, which can be kept private or shared publicly with the user community. A week-long deployment study with 15 participants showed that users were able to easily add and curate their own content and locate help resources generated by other users. They found the social curation approach to be helpful in a variety of contexts, and the majority of users wanted to keep using the system beyond the deployment. We discuss the potential of Social CheatSheet, as an application-independent platform driven by community curation efforts, to lower the barriers in finding relevant help and instructions for feature-rich applications.

Acknowledgements

I would like to take this opportunity to acknowledge and thank several people who have generously helped me to succeed in the course of my studies and thesis writing.

Above all, I owe immense gratitude to my supervisor, Dr. Parmit Chilana, for her continued support, guidance, and encouragement throughout the past few years, without which this thesis could never have taken shape.

I would like to thank my co-supervisor, Dr. Daniel Vogel, for his valuable advice and for stepping in to help out with the unusual situation of both of my supervisors leaving the school. I would also like to thank my former co-supervisor, Dr. Michael Terry, for his many contributions to this research and all the conversations that helped to refine its focus and direction.

Thank you to my thesis readers, Dr. Mark Smucker and Dr. Mei Nagappan, for their important insights and feedback for improving the clarity of this thesis.

To Nathaniel Hudson, Anson Ho, Shruti Dembla, Celena Alcock, April Wang, and other fellow lab members at both UW and SFU, thank you for all your help with various research projects and all the fun times we've shared in between.

I also wish to thank Dr. Mark Hancock, Dr. Stacey Scott, and everyone in the EngHCI group for all the valuable feedback you've offered in lab meetings and elsewhere, and for providing such a welcoming and intellectually engaging environment to discuss research.

Finally, thank you to Samantha Marks, Max Marks, Gayle Marks, Paul Vermette, and Fairman Vermette for all the love, support, and patience you've offered in countless ways throughout my education.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

Using an unfamiliar feature-rich application can often be daunting for users. Although the web offers thousands of help resources, locating the most relevant and useful instructions pertinent to a task can be frustrating and time-consuming. Instead of sifting through verbose documentation, lengthy text and video tutorials, blogs, wikis, and other help pages, many users instead prefer social forms of help, such as asking a question over email [40], posting questions on their social network [36], or seeking help from software Q&A forums [32,46]. Through these social means, users can obtain direct task-relevant curated instructions from other users or experts, such as a paraphrased explanation, an annotated screenshot, or a snippet of content extracted from an existing help resource (e.g., a highlighted tutorial step, or a relevant video frame).

Despite the benefits of curated help, the actual curation process can be cumbersome for the help provider, as multiple tools are required to capture the screen, annotate, save, and share instructions. Furthermore, these help snippets can be difficult to re-discover once they are buried in a private inbox, a messaging service, or a Q&A forum thread [24]. Recent developments in commercial and research applications for note-taking and personal information management (e.g., Evernote [16], CheatSheet [50]) offer the possibility to streamline the process of taking screenshots, adding annotations, and saving help content for personal reuse. Tools such as CheatSheet [50] further provide mechanisms for users to easily retrieve their own annotated “cheat sheets” as an overlay atop the relevant application. What if these personal cheat sheets and notes could be augmented with content from other users learning or troubleshooting the same application? How would users want to contribute and locate relevant content? In what contexts would this approach be most helpful?

In this thesis, we investigate the concept of social curation of software help content by developing a novel platform, Social CheatSheet, which allows multiple users to curate help content and collaborate. By using Social CheatSheet, users can automatically discover relevant community-curated instructions, tips, notes, and tutorials within any web application, and can easily curate and share their own content.

Although Social CheatSheet takes inspiration from several existing HCI and CSCW approaches, such as community-based help systems [7,29,30,33] and social annotation techniques [15,23], our approach is unique in that 1) it allows users to easily generate and aggregate task-focused curated instructions and multi-step tutorials using a combination of their own annotated screenshots and snippets of web-based help resources; and, 2) it makes it possible to leverage the expertise of the user community to quickly find an answer to the question *"What is the best instruction or tutorial for me to learn task X in this application?"*. Importantly, users can retrieve and edit other users' curated help on any web application by installing a browser extension, bypassing the potential barrier of relying on application owners to integrate the community help system [6].

To inform the design of our system, we first conducted two formative studies with 10 users each. In the first study, we observed how users currently interact with web-based resources as they work with an unfamiliar feature-rich application. In the next study, we developed a minimal prototype using our standalone CheatSheet [50] idea with pre-populated content to understand how users would consume and perceive cheat sheets curated by other users. Accordingly, we designed Social CheatSheet to facilitate community-based curation and retrieval of curated content that matches users' tasks, expertise, and learning preferences.

To evaluate the concept of social curation, we carried out a week-long deployment of Social CheatSheet with 15 users. We elicited several insights into how users perceived the curation process, how they viewed notes and tutorials created by other users, and which aspects of Social CheatSheet were deemed most beneficial. The majority of users felt favourably towards the system, including wanting to keep using the system beyond the deployment. Despite a small community of users and limited help content, two-thirds of users revealed that they discovered something new during the deployment period just by browsing other users' curated content. In terms of contributing content, a third of the users said they would likely not add their own notes and tutorials. But, given the ease of using the curation features, over half of the users were neutral and not opposed to adding content if the community were to grow in the future.

We discuss the potential of Social CheatSheet as a platform to help lower the barriers to using feature-rich applications and opportunities for future work to incentivize community participation and innovate in personalizing and recommending targeted curated help content. As such, our vision is for Social CheatSheet to serve as a modern version of the minimal manual [5] that does not require users to leave their current tasks, does not need involvement from the application owners, and allows users to retrieve relevant task-specific instructions curated by the community.

The main contribution of this thesis is the design of Social CheatSheet, a novel easily-accessible platform that can serve as a modern community-generated minimal manual [5] for learning unfamiliar tasks within a web application. Additionally, our field study contributes novel insights into how such a community-curated help platform can be used to add and retrieve content in the context of using a real-world application.

This thesis is structured as follows: Chapter 2 outlines a variety of related work, including systems and software tools that have inspired some of the design elements of Social CheatSheet, and how our work differs from them. In Chapter 3, we discuss two formative studies that we conducted to gain a better sense of the behaviour of software learners and what their specific needs would be in a community-based software help overlay. Chapter 4 details the user interface of our new Social CheatSheet system and how its interface design addresses the needs discovered in the formative studies, while Chapter 5 goes through the more precise implementation details that form the core of the system. In Chapter 6, we outline the protocol and results of a week-long field deployment that we conducted to evaluate Social CheatSheet in a more realistic environment. Finally, Chapters 7 and 8 discuss other implications from this research, potential future work, and the conclusions that can be drawn.

Chapter 2
Related Work

Our Social CheatSheet system builds upon and complements a number of previous innovations in HCI, including advances in community-based software question and answer (Q&A) systems, recommender systems, annotation tools, and help re-finding tools. Although Social CheatSheet extends the core idea of the standalone CheatSheet system [50], it contributes a new collaborative system and novel insights into designing community-based means of creating and browsing tutorials, instructions, tips, and other types of software help content.

Community-based Q&A Systems

Social help in the form of community-based Q&A has become increasingly common over the years, with well-established communities for both open source software and commercial products in the last decade [30,32,45,47]. Our in-application social help approach is most similar to systems such as LemonAid [7] and IP-QAT [33], which focus on delivering community-driven Q&A-style help based on application context. Like these systems, our motivation was also to: 1) reduce the back and forth between different applications and web help resources that users often experience; and 2) allow the user community to leverage each other's experiences and expertise.

However, unlike these Q&A tools, Social CheatSheet's primary focus is on helping users locate curated task-based visual instructions and step-by-step tutorials available on the web or created by the user community. Furthermore, Social CheatSheet users can customize the retrieved content by filtering by task, expertise, and learning style preferences and have access to a personal space for bookmarking useful content or adding their own private curated content. Finally, since Social CheatSheet is a browser extension that anyone can install, it does not have to be adopted or integrated by application owners, which is known to be a potential barrier [6] for embedded community help systems.

Community-based Software Tutorials

Another class of research has explored how the user community can be involved in enhancing tutorial content to help users make better use of unfamiliar features and reach new skill levels when learning software. For example, the CADament [31] system leverages the idea of gamification to engage users in a multiplayer tutorial system and promote collaborative learning experiences for a CAD application. Despite the benefits of software tutorials, users often have difficulty locating relevant content within tutorials and it has been shown that users can benefit from community-based annotations that highlight key steps [25] or comments that tightly integrate with different parts of the tutorial [4]. Users can also benefit from seeing alternative or additional demonstrations and explanations in video tutorials [29].

Although Social CheatSheet builds on the concept of community-authored tutorials, it does not generate video or documentation-style tutorials (based on feedback from the formative studies); rather, it offers a unique tutorial authoring feature to segment a task as a sequence of individual annotated step-by-step screenshots and allows users to attach external help snippets (e.g., a video frame, FAQ, or forum answer) to enhance the content. These tutorials are tightly integrated within an application and allow users to focus and easily navigate between individual curated tutorial steps without having to consult external resources. Furthermore, any future user can add, extract, or edit any part of the tutorial to improve the instructions for her own use or for sharing with the user community and can personalize the retrieval of tutorials by specifying different tasks, level of expertise, and learning preferences.

Lastly, a number of industry solutions such as *StackOverflow* [55] and *Quora* [56] have risen in the past decade as attempts to provide canonical Q&A resources on the web. In particular, *StackOverflow*'s sister site *SuperUser* [57] focuses primarily on building a collection of community-generated questions and answers about general software usage. However, these standalone Q&A resources still require users to actively interrupt their task to visit external websites and search engines to find solutions to their problem. Social CheatSheet addresses this by automatically retrieving community help content related to the current web application, all presented within the application itself, so that users can more easily incorporate the social help-seeking process into the natural flow of their application usage. In particular, it includes a "drawer" interface on the side of each web application, allowing help resources to be accessed without any task switching or significant interruption.

Community-based Software Help Recommendations

Several HCI systems have investigated the idea of recommending help based on past actions of other users. For example, the HelpMeOut [22] system suggests solutions to help debug error messages based on what other programmers have done in the past. Similarly, CommunityCommands [34] contributes a collaborative filtering algorithm that can make new command suggestions based on actions of other users in a CAD application. More recently, the DiscoverySpace system [18] explores the idea of suggesting task-level actions to Photoshop users by harvesting data from user communities.

While Social CheatSheet captures basic user actions (e.g., clicking, scrolling, typing) for generating tutorial steps, it does not detect individual commands or application-specific features. Furthermore, none of these low-level interactions are stored or used later to affect the retrieval algorithm. To retrieve relevant content, Social CheatSheet instead relies on task tags and explicit contributions from the user community (e.g., upvotes, downvotes, views, and other metadata). Additionally, when someone views a note or a tutorial step, Social CheatSheet can suggest notes with related or alternative explanations on the same topic, or lead users to the next step. These recommendations are based on the visual and textual similarity of the content and the related metadata, not analytics collected about user activity.

Tools for Augmenting Web Page Content and Functionality

Although not in the domain of software help, many HCI systems have explored how users can augment web page content and functionality and even share these augmentations with other users. Some of these systems [15,23] add rich social text annotations, such as comments and mark-up, as a layer overtop different elements of a web page. However, these systems largely employ page-based annotation tied to the content of the page (e.g., sentences and paragraphs). In contrast, Social CheatSheet uses a task-based annotation approach: instead of attaching annotations to specific page elements, it inserts a library of annotated screenshots and snippets tied to different application subtasks, generated and curated by users or extracted from existing web-based help content.

In addition to page-level annotations, some systems have explored ways of collecting and summarizing web-based information related to a topic using automatic approaches and templates [14] or using socially annotated schemas [26]. Social CheatSheet has a similar goal of consolidating and organizing content from a variety of existing web help resources, but these resources need not follow similar templates or schemas to be incorporated into user-generated tutorials. Instead, Social CheatSheet is driven more by a deliberate community effort to curate and improve a shared, growing collection of software help tutorials than by its ability to automatically capture web-based content.

Other ways of augmenting web pages include systems that allow users to share detailed browsing activity with friends and even sync browsing sessions (e.g., PlayByPlay [52]) or adapt the structure of specific pages based on community redesigns (e.g., CrowdAdapt [39]) and accessibility needs [10]. However, Social CheatSheet aims not to interact with its underlying web page functionality or adapt the interface, beyond inserting its unobtrusive, initially-hidden layer of help resources. The main goal is to make it easier for users to discover curated help content for the application currently being used.

The visual annotation features in Social CheatSheet are most closely related to *ScreenCrayons* [41], where users can add notes and highlights to any captured screen. However, the ability to annotate screens is not one of our primary technical contributions, and although our system relies on these annotation features as a backbone, its core contributions lie instead in the community curation of in-application software help and the application-specific linkage of scattered web help content into a single resource for software learners.

Finally, there are a number of other commercial tools, such as *Evernote* [16] and *OneNote* [58], which support general note-taking on the web and offer several features to clip and annotate sections of web content. Our system differs from these tools in several key ways: 1) Social CheatSheet relies primarily on the visual representation of web applications as the default annotation platform to facilitate recall among software learners, 2) Social CheatSheet allows users to quickly generate and publish entire multi-step tutorials for a software task by monitoring users' page interactions, and 3) Social CheatSheet automatically retrieves a selection of community-curated notes and tutorials for the current application, functioning more like a collaboratively evolving "cheat sheet" for the application, rather than a generic personal notebook.

Tools for Re-finding Help Resources

Another emerging class of systems has focused on re-finding previously found useful help. For example, *InterTwine* [17] supports task-based re-finding by tying together actions in a software application with browser histories and search queries. This and other systems such as *SearchBar* [35] assist users with continuing or re-learning a previous task based on a history of resources they have already used, but these approaches typically assume the use of a general search engine as the primary means of finding information.

Social CheatSheet instead tries to facilitate a wider range of users' needs, allowing useful information to be bookmarked independent from the browser, and supporting a more contextual mode of help-seeking based on the application being learned and the tasks being performed within it. Additionally, its social features make it easier to see what resources other people have found helpful for a given task, while still providing space for one's own personal notes. The original CheatSheet [50] system (described below) was also designed to minimize the effort of re-finding help, although the Social CheatSheet system extends this concept in several other ways beyond simply re-finding, discussed below.

The Original CheatSheet Concept

Social CheatSheet uses the backbone of our previous system CheatSheet [50], which was designed for users to save screenshots and snippets of help content as individual "notes" atop any web application, functioning as a personal memory aid for software learners. Past studies with CheatSheet [50] have indicated that a natural extension would be to explore community-based re-finding of help content. But how a community-based re-finding and sharing platform could be designed and whether users would even find it useful in the context of their learning tasks was not investigated—this is a key focus of the studies in this thesis. In particular, Social CheatSheet extends the in-application memory aid concept [50] by connecting potentially many users of the same applications through a shared community-curated “cheat sheet,” which they all draw from and contribute to atop those applications. We will briefly outline the original features of CheatSheet that were later extended to fit the needs of our new system.

Adding Visual Annotations and Metadata

In the original design of CheatSheet (seen in Figure 1) a browser extension would capture a screenshot of the active page when the user decides to add a new note. This would open an overlay on top of the page, showing the screenshot within an editable canvas. From here, the user can crop and mark up the screenshot with a number of standard annotation tools, such as a highlighter, arrows, boxes, and text.

Alongside the annotation canvas, users can also add textual descriptions to their note,

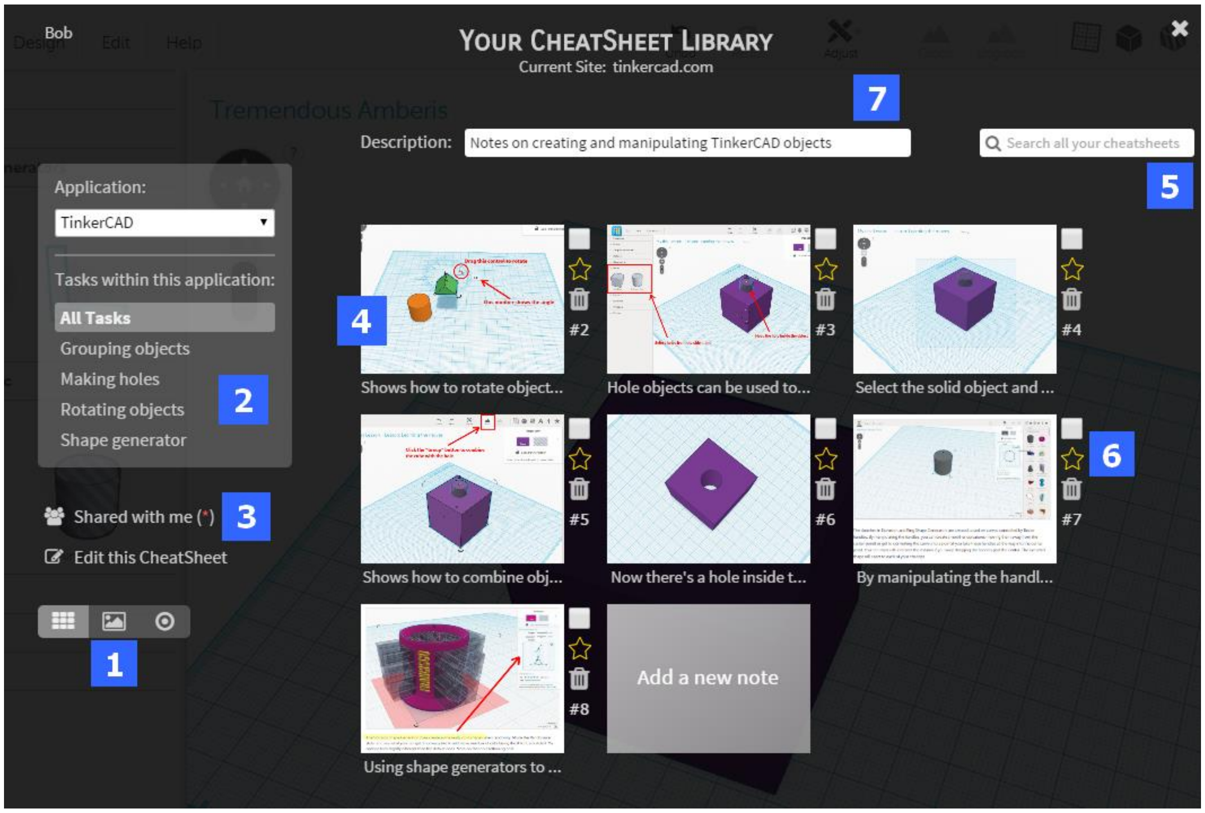

YOUR CHEATSHEET LIBRARY
Current Site: tinkercad.com
Description: Notes on creating and manipulating TinkerCAD objects
Search all your cheatsheets
Application:
TinkerCAD
Tasks within this application:
All Tasks
Grouping objects
Making holes
Rotating objects
Shape generator
Shared with me (*)
Edit this CheatSheet
Shows how to rotate object...
Hole objects can be used to...
Select the solid object and ...
Shows how to combine obj...
Now there's a hole inside t...
By manipulating the handl...
Using shape generators to ...
Add a new note
#2
#3
#4
#5
#6
#7
#8
1
2
3
4
5
6
7

Figure 1: The original CheatSheet library view, showing notes about the TinkerCAD 3D modeling application. Here, users can (1) switch to different views of the content such as a carousel or a radial layout, (2) filter the results by certain tasks, (3) view notes that have been specifically shared with them by another user, (4) zoom into notes in-place or click to view them in the larger editor, (5) quickly search their entire library for a particular keyword, (6) delete notes or mark them as favourites for ease of re-finding, and (7) add overall descriptions for an entire task. (From Vermette et al. [50])

which can be automatically sourced from the underlying page by having the desired text selected before the screenshot is captured. Upon saving the note, the URL of the underlying page is saved as well, and can be retrieved later to serve as a page bookmark. This may be useful if the note's content is from an external help resource rather than the application it describes.

Aggregating Relevant Help Content

Since the notes stored in CheatSheet may be aggregated from many scattered web resources, CheatSheet lets users tag notes with both the name of the application and any specific tasks within that application that the note relates to. When entering tags, CheatSheet will automatically suggest any tags that are already attached to notes in the user's library. Later on, these application and task tags can be used for filtering the notes that CheatSheet retrieves and displays in the library view.

Insights from CheatSheet's Evaluation

Although CheatSheet originally included a simple means of sharing content with other users, this was limited to sending a single note to only a single person at a time, provided their username or email was known. However, usability testing of CheatSheet indicated that users were highly enthusiastic about seeing more social features and being able to make their notes available in a more public setting. For example, some participants saw potential for IT support teams to use CheatSheet as a platform for publishing visual FAQs or other software instructions. Social CheatSheet addresses this, first and foremost, by making all notes and tutorials public by default, to encourage the creation of a community help resource rather than simply a personal memory aid. It also adds features to promote discussion and collaboration on notes and tutorials, as well as a voting system for users to provide feedback on the helpfulness of content, which can later be used to retrieve content based on its popularity.

Chapter 3
Formative Studies

To derive initial design requirements for our social curation platform, we carried out two-formative studies. We first examined how software learners make use of web-based help resources when learning a new feature-rich application. In addition, we also investigated whether users would find it useful to have overlaid help content created by other users of the same application, using a minimal prototype similar to CheatSheet [50], but modified such that users could see all of the notes created by others.

Overview

We carried out two observational studies with a common scenario to observe help-seeking behaviour using two different mechanisms. Our test site for this study was the web-based *YouTube Video Editor* [59], which offers a number of features for creating, editing, and publishing videos online. We selected this application because it allowed us to create engaging tasks that users could relate to, while involving a range of both basic and more challenging features to learn and use.

Study 1 (Web Help): Our goal with the first observational study was to gain qualitative insights into the breakdowns that users experience when learning a new application, and how they make use of existing help resources on the web to resolve those breakdowns.

Study 2 (CheatSheet): The goal of the second study was to determine to what extent users find it helpful to see other people's help content overlaid publicly within the original CheatSheet prototype, and to derive design and feature requirements for our community-curated in-application help overlay.

Participants

We first obtained ethics clearance for our study protocol, and then recruited 20 participants for the formative studies – 10 participants (5M/5F) for the Web Help study, and 10 more participants (4M/6F) for the CheatSheet study. All participants were between the ages of 19-64, and the median age range in both studies was 25-34. All participants had completed at least high school education. Participants were recruited through mailing lists, word of mouth, and flyers placed on bulletin boards at the University of Waterloo. Only one of our participants (for the web help study) had used the video editor before, but said they had no experience with any of the specific features we were asking them to use in the scenario. For participating in the study, participants were each remunerated with a $15 Amazon gift card.

Scenario and Tasks

In both studies, participants were asked to perform the same series of tasks (see Figure 2) in the YouTube Video Editor. Participants were asked to imagine a scenario where they were returning from a vacation and wanted to put together a short compilation of their best vacation video clips and pictures prior to catching their flight home. In the scenario, their flight would be leaving in 30 minutes, and they wanted to try to finish the video before then.

First, they were to upload two short video clips and an image to the editor from the study computer, and then combine these into a single video that included animated transitions between the three sections, a text banner title, "speech bubble" annotations at two points

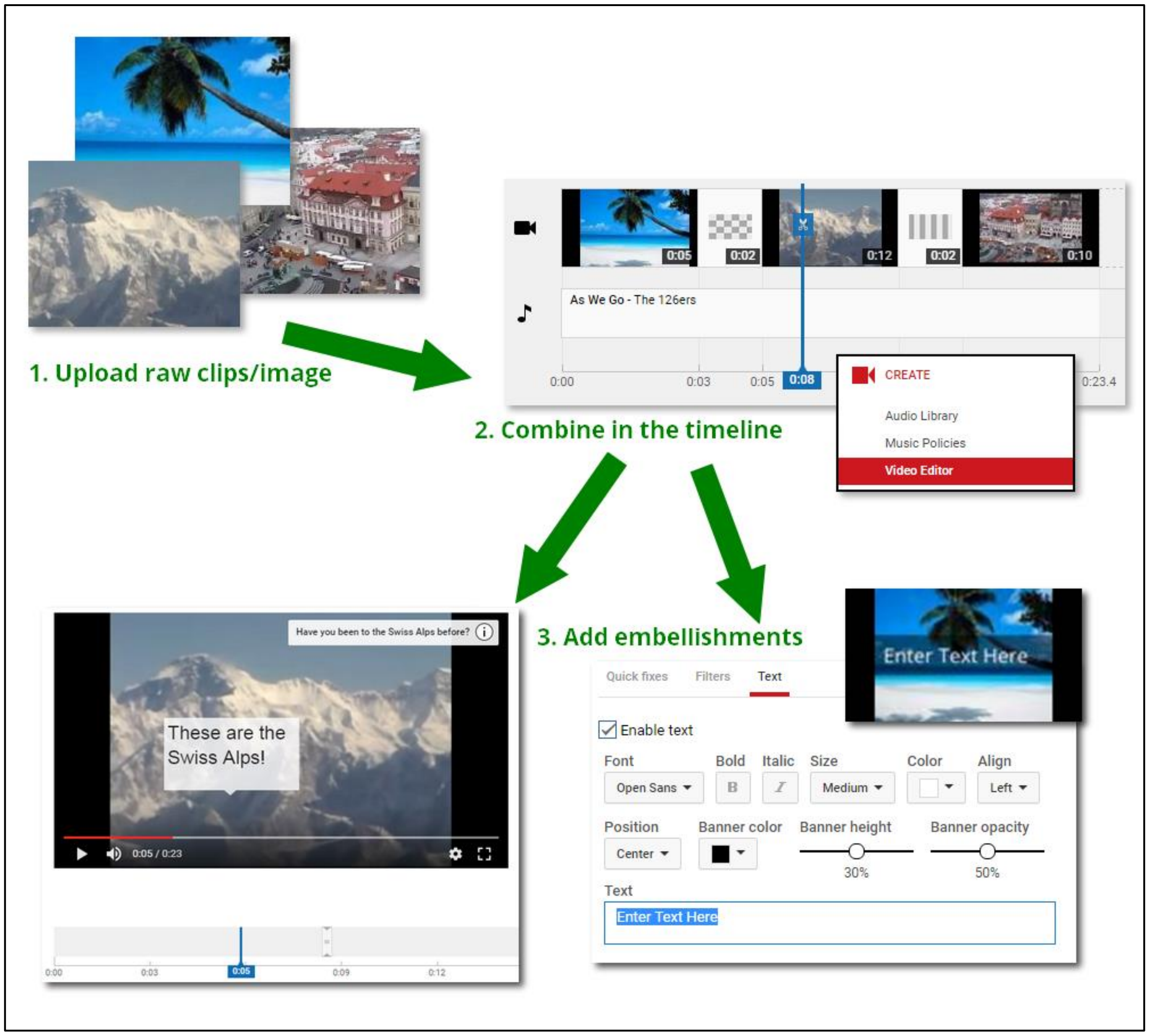


Figure 2: Participants were tasked with uploading two video clips and one image to the YouTube Video Editor, combining them into a single timeline with transitions and music, and adding further embellishments such as annotations, banner text, and embedded polls.

during the video, and interactive poll cards that would pop out from the corner of the video and ask viewers whether they had visited the places being shown. Finally, we also asked participants to add a specific music track from the YouTube editor's library. The scenario specified that the finished video should be sharable with friends and family via a link, but not listed publicly.

We provided participants in both studies with a written version of the scenario outlining the tasks above and asked them to attempt as many of the steps as they could in the 30-

minute time slot. We opened the video editor's Dashboard page in the active browser tab at the beginning of the study, logged into a fresh YouTube account made solely for the purpose of this study. Finally, in a separate browser tab, we provided an example video displaying the "model outcome" with all of the features described above, which the participants could refer back to at any time to compare their progress with it.

We asked participants to think aloud as they performed the task, while one researcher observed and noted where participants encountered breakdowns or sought help. We also captured screen recordings during the task, to corroborate our observations. Following the 30-minute task, we conducted a 10-15 minute semi-structured interview with each participant, asking about their experience using the video editor, their normal help-seeking habits when learning to use software, and (if applicable) their thoughts about the CheatSheet prototype.

We measured the time it took participants to complete the entire video editing task, with an upper limit of 30 minutes (after which they were stopped). We also measured the amount of the task that each participant completed, as a percentage. Participants who managed to complete all aspects of the task received a completion score of 100%, while who failed to complete the entire task in the allotted time were assigned a completion score below 100%, determined by how many of the sub-tasks they successfully completed (each given an equal weight of 10%):

1. Upload the video clips
2. Upload the image file
3. Combine the video clips and image in the video editor's timeline
4. Add transitions between the three sections
5. Add banner text to the first section
6. Add the audio track to the timeline
7. "Create" the full video
8. Add annotations to the created video

9. Add poll cards to the created video
10. Publish the video with a privacy setting of "Unlisted"

Web Help Study: While completing the study task, participants in the web help study were given the opportunity to seek help on the web using whatever strategy they were used to (e.g. using a search engine, consulting built-in help manuals, or finding tutorial videos). These participants were not given access to the CheatSheet prototype during the study.

CheatSheet Study: Participants in the CheatSheet study were given a brief verbal tutorial demonstrating the CheatSheet help system prior to beginning the task. They were asked to use only our CheatSheet prototype to find help when they needed it during the study, and were restricted from using any other means of seeking help.

Three members of our research team prepopulated the CheatSheet prototype with several explanatory notes on using the video editor, based on a variety of web-based tutorials, Q&A sites, help videos, built-in help resources, and personal knowledge of the editor. All of these notes were made visible in the prototype as if they were coming from a larger community of contributors. We ensured that every component of the study task was explained in one or more of these notes and, for completeness of the help content, included additional notes on other major features of the YouTube editor as well.

Key Findings

Since all the participants were new to using the features of the YouTube Video Editor that we tested, we observed in both studies that they frequently struggled to locate and understand the relevant video-editing functionality for the task, and all participants ended up seeking help with multiple parts of the task.

In particular, the initial step of uploading the video clips was one of the most frequent breakdowns faced by participants in both studies, since it first required them to discover a small, undistinguished "Upload" button in the top right corner of the editor page. There were several different breakdowns experienced with this step, such as attempting to drag video files directly from the filesystem into the editor (which would simply cause the browser to play them immediately), attempting to upload the image file through the video uploader (upon which the editor would prompt them to create an image slideshow), and directly publishing the individual video clips separately once they had finished uploading (which was unnecessary given the overall goal of combining them into a single video).

Another frequent point of difficulty was a failure to recognize that the combination of the video clips and title image must be "created" as a standalone video before the option to add annotations and poll cards would become available. Furthermore, among the participants who succeeded at that step, many still struggled to figure out where in the editor they could add annotations and polls, in part due to a lack of correct vocabulary. Some participants were not familiar with the term "annotation" with regard to YouTube videos, and some did not recognize that polls are one of several varieties of "cards" in the editor.

Web Help Study

We observed that all but one of the participants in the web help study relied on Google search to find help. One common observation was that users were frequently switching between several browser tabs for the video editor pages and the help resources they had found. Participants would often visit several different resources from their search results and then leave each of them open in different tabs, as well as opening different video editor pages in different tabs (whether intentional or not). This behaviour would quickly lead to a seemingly confusing buildup of tabs. In many cases, participants would appear to lose track of which tabs they were editing the video in, which tab contained the model solution video, or which one had a certain help page they were working through, leading them to click through

All participants either clicked on the built-in help pages or issued queries on Google, making at least one textual query leading to search results, with an average of 4.2 search queries per user. These queries commonly included terms and phrases, such as *"youtube* *editor upload"* or *"youtube add photo"* and often ended up at web pages which were not directly relevant to the task. A recurring example of this was a page in the official YouTube help manual titled *Upload audio or image files* [60], which 8 of the 10 participants visited. Despite the promising title, this page only detailed ways to convert a series of images into a "slideshow" video, or upload a single audio file as a standalone video, but was not helpful in explaining how to upload these files in such a way that they could be further combined, as required for the study task. This appeared to cause many of the participants some degree of confusion, as they tried to follow along with these instructions but would end up stuck on the wrong page, in the wrong editor state, or with results that didn't resemble the model solution.

> *"Reading the instructions that it gave me, it just told me to go to the corner, upload, and go to what I think I'm supposed to be in, but I don't even know if I'm in the right window anymore." (WH02)*

Overall, participants in the web help study spent an average of 28.9 minutes (s.d.=2.33) working on the task, with only two participants completing 100% of the task within the 30-minute time limit. On average, participants completed 63% (s.d.=35.6%) of the task. The behaviours of flipping back and forth through tabs, wasting time on unrelated help resources, and sifting through several search results to find relevant content were likely major contributors to low completion rate and the long time it took these participants to complete the task.

Another key observation was that some of the participants had a clear preference for either reading through written tutorials or viewing instructional images and videos, and they indicated that their typical help-seeking habits tend to focus more on the type of media they learn better from:

> *"I know there are always YouTube videos and tutorials. I feel like it's more helpful when I see it or a video instead of reading something, because I'm like a visual learner. So I watched the first video and it was really helpful." (WH03)*

"I don't use videos, because I have to spot the point where exactly the instruction I need is located, and I don't want to have some boring introduction of somebody saying 'Hey, hi, my name is, whatever I'm going to show you…' – you don't know how long that lasts. So locating the information that I need in a video is time-consuming, when I can instead just scan the text and see a list of steps and this is what I need. I don't want to go through a six-minute video to find a very specific piece of information. So I very rarely use videos, unless it's the only means available." (WH05)

In some cases, participants said that they might prefer particular modes of help over others depending on the nature of the problem:

"For something like this, this was a UI problem, where I didn't know where the button was that I needed to click, so a video makes much more sense in that case. The thing with the video is, at the very least you can match it through the UI… For something like this, that's very UI-dependent, the video's probably going to be faster." (WH04)

"I think when there are a lot of steps, then videos are good… If there's a particular way that you have to do something in order for it to work, then that might come across with just a written description." (WH06)

CheatSheet Study

In the CheatSheet study, we found that participants took 27.9 minutes (s.d.=2.88) on average to complete the task, with 4 out of 10 completing the entire task, and the rest being stopped at the time limit. Despite this being rather slow, participants using CheatSheet also managed to complete 80% (s.d.=24.0%) of the task on average, a promising result given that they were unable to access their normal means of seeking help and had to rely on an unfamiliar tool to learn about the editor's features.

One of the most common breakdowns experienced by the CheatSheet users was a failure to understand that the combined video must be "created" within the YouTube editor before it can have annotations and polls added to it. Prior to this step, it only exists as an editable timeline, but not as a standalone video to which these elements can be attached. Although one of the prepopulated notes in CheatSheet did contain this information, participants rarely viewed it because they were searching for information about the process of actually adding annotations and polls, not realizing that they had missed a crucial step. This would then lead to them spending time searching in vain for certain menus and editor pages that were not yet available to them.

To find help from our prototype, participants clicked on the pre-populated task tags about 5.9 times on average and issued 4.6 queries on average using our built-in keyword search feature. Although our participants were using our prototype for the first time to learn an unfamiliar application, it was encouraging to see that they found it helpful overall to see other users' curated cheat sheets overlaid within the application. They cited various reasons, such as being able to access help from within their application, without interrupting their train of thought by visiting another web site:

> *"The best thing about it [CheatSheet] was the overlay so I didn't have to interrupt my train of thought to go on Google and search for something most relevant…I could just pop it up and type in a couple of keywords and see if I find anything relevant to what I was looking at." (CS03)*

Many participants also felt that being able to see task-relevant help content right away was an improvement over struggling with conventional search results:

> *"CheatSheet already had the help I needed, so it was easier because it had pictures which pointed me where to go… it had specific answers which I was looking for, compared with Google where I might not come across the most relevant answers right away." (CS08)*

Other participants found the visual nature of retrieving help through the CheatSheet prototype to be much more helpful than strictly text-based help resources such as FAQs or lists of instructions:

> *"I really like the way it [CheatSheet] had image thumbnails on everything, because that way it was a lot easier to tell if something was relevant to me or not, because I could tell if I was looking at the correct screen or not before opening up an answer related to that screen." (CS06)*

"It [CheatSheet] is more visual too because usually FAQs are different, they are long lists of questions. But this has images and screenshots, so you can just click on it and things show up so you don't have to navigate to different pages to find help." (CS07)

However, it was common for participants to struggle with moving between different sub-tasks or completing some of the more involved sub-tasks due to a lack of guidance regarding "what to do next" when multiple steps were involved. This supported our earlier observation that a key breakdown was occurring when participants unknowingly skipped a mandatory step in the video editing process. Some mentioned that they prefer to have all the steps laid out in order, or some other "big picture" overview:

"At least for me, [what I need is] the 'first thing you need to do'. Maybe one paragraph that says 'this is the process you use to create a video'... four lines, say. That's actually a useful summary, and now I can actually go in and figure those things out. My feeling is if there's a 'do this first, then read this', it might be more helpful." (CS05)

Some participants were enthusiastic about the social nature of a public "cheat sheet", particularly the ability to leave input on which help resources they found the most useful, to assist future users with the same problem:

"Having a kind of useful place where you can up vote things prominently if you find it useful, so you can bring it higher without needing to have Google pick up on the fact that it was a good answer for people; that would be very useful." (CS01)

"I feel like if someone is looking for an answer, I think looking at number of views or popularity would help them make a decision on which one to look at first." (CS08)

However, others raised concerns regarding the privacy of information captured by CheatSheet, and whether this would lead to them or others unintentionally revealing sensitive information:

"Since they made this so everyone sees it... Doesn't it mean I might see someone's account numbers or passwords in [the screenshot], that they put in by accident?" (CS04)

Furthermore, although participants overwhelmingly found CheatSheet easy to learn and enjoyable to use, a few of them did note that its keyword search functionality felt too limited compared to modern natural language search engines:

> *"I tried typing in all sorts of things, like 'uploading a youtube video' was saying 'No notes', but later on I saw a note in here about uploading… It didn't usually understand what I wanted, the way Google does." (CS05)*

Lessons Learned and Design Implications

Based on the main findings reported above, we now reflect on some of the implications for the design of our community-curated software help overlay. In particular, the Web Help study aptly demonstrated that despite the abundance of help resources available on the web, many of the top results returned by a search engine may be of limited use for a given task, and can potentially cause further difficulties if their applicability is poorly understood. The CheatSheet study further brought to light several limitations of the current system and inspired several changes that could improve its ability to serve as a viable help resource.

It was also surprising that the majority of the participants failed to complete the entire task within the 30-minute time limit. In retrospect, it may have been beneficial to provide more time or to remove some of the sub-tasks to give participants a better chance to finish, as this may have resulted in more robust quantitative data regarding task completion and efficiency. However, we also noted that a large number of our participants seemed frustrated with the video editor by the end of the task, and several struggled even to complete the first sub-task of uploading the video clips, a fact which might discourage a longer study.

While the overall response towards our vision of a social help platform was positive, user feedback indicated that the experience of discovering and using curated content was currently not adequate and could be improved in several ways. Through affinity diagramming, we distilled these areas for improvement into five key design goals.

Community-oriented features: Users were enthusiastic about participating and learning from the community, which could be achieved via features such as voting on the helpfulness of notes, seeing the most popular content, receiving recommendations for related content, adding comments, and requesting clarifications on a problem.

Step-by-step tutorials: A common point of feedback from the participants was that viewing individual notes was not always sufficiently helpful when a task required multiple steps. Since the notes in our CheatSheet prototype tended to describe discrete, separated subtasks like adding an annotation to a video, some participants felt that they were lacking the "big picture" or were unsure where to go next after completing a step. Several participants who used CheatSheet requested the ability to view help content in more connected step-by-step tutorials.

Personalization options: Participants talked about how they wanted a quick way to see help that was comprised of either more visual or more textual explanations, based on their learning preferences and the problem context. Similarly, they noted that some help seemed to be targeted towards more experienced users, while they wanted to see something relevant to them as a first-time user. Our test prototype did not offer a way to browse or filter based on content type or intended audience, and participants noted that this was a limitation of other modern help systems they use that adopt a "*one-size fits all*" approach for providing help.

Flexible natural language search options: A key problem for all users in the CheatSheet study was using our prototype's rudimentary search feature. Every participant tried to search by keyword at least once, and they frequently indicated the current search mechanism to be too restrictive. Participants noted that as the size and scope of curated help content grows, a more flexible natural language search engine would be necessary to locate relevant content.

Privacy controls: Our participants commented on the possibility of seeing someone's private information (e.g., banking details, passwords) by mistake, if it appeared as part of the help content. This was also a concern about participation in the curation platform in the future: users would need to be able to mark things private when necessary or hide specific sensitive content in a screenshot.

In summary, we learned that users were positive about using help curated by other users in the context of learning a new application, but some key interaction capabilities would be needed in our social platform to adequately foster community-based curation and retrieval of relevant content.

Chapter 4
User Interface Design

Based on the findings from our formative studies, we designed Social CheatSheet, a novel social platform that allows many users to curate, collaborate on, and discover relevant instructions, tips, and tutorials atop any web application. Our platform is built upon the basic functionality of the personal note-taking application CheatSheet [50], extending it with several new features to accommodate community participation and more targeted searching, browsing, and filtering options.

Adding Curated Content

Extending the above CheatSheet features, Social CheatSheet offers a number of additional and improved ways for users to add and curate content. By default, any note or tutorial added to Social CheatSheet is public (unless explicitly marked as private). Any content can be edited by any user in a wiki-style approach, with the system keeping track of which user contributed edits to a note or tutorial.

Adding Notes

The new note creation workflow in Social CheatSheet builds on the original concept with some additional improvements. When a user clicks "Add a Note" (Figure 3.1 or 5.1), in addition to capturing the screenshot and opening it in the "curation interface" overlay,

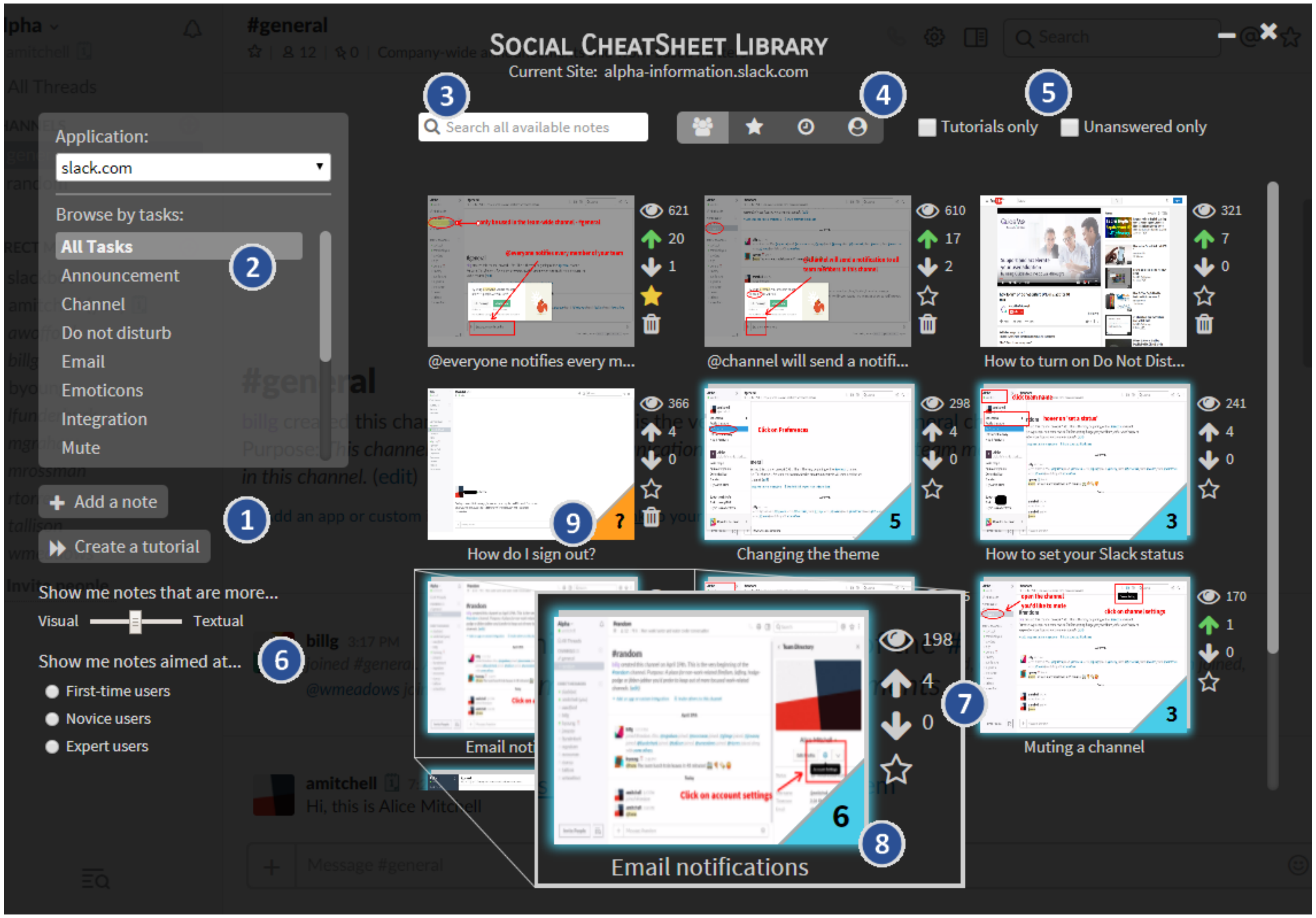
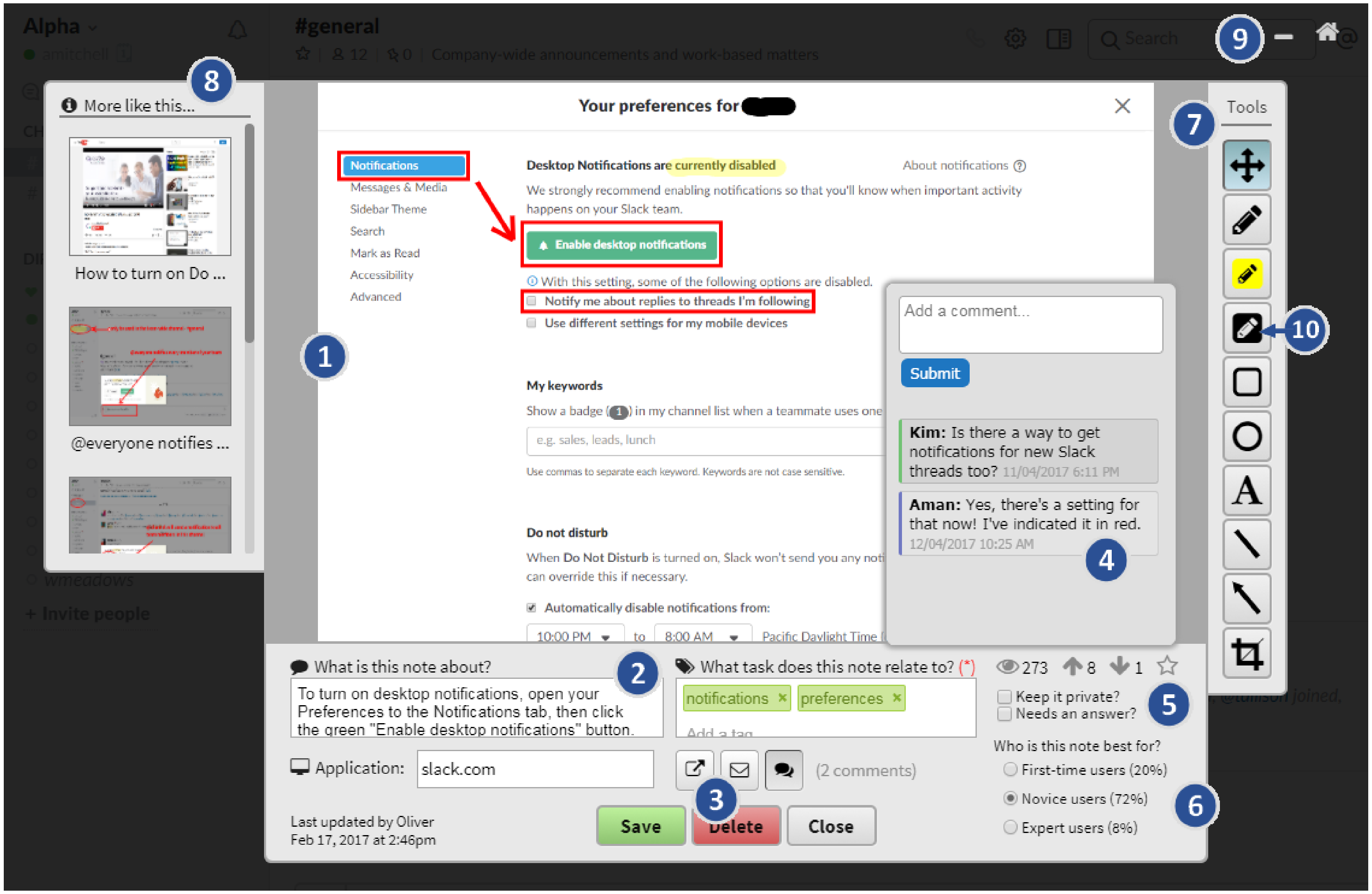


Figure 3: The full Social CheatSheet library view, in which all of the curated help content relevant to the current application is listed. Users can (1) easily add their own curated notes or tutorials, as well as filter content by (2) existing task tags or (3) natural language search queries. They can (4) sort the display by popularity, date, owner, and bookmarks, (5) limit the display to only tutorials or unanswered questions, and (6) further filter the content by changing their learning style preference and expertise level. Each note or tutorial shows (7) the number of views and votes, and users can easily identify (8) tutorials by a blue triangle and (9) unanswered questions by an orange triangle.



Your preferences for [redacted]

Notifications
Messages & Media
Sidebar Theme
Search
Mark as Read
Accessibility
Advanced

Desktop Notifications are currently disabled

About notifications

We strongly recommend enabling notifications so that you'll know when important activity happens on your Slack team.

Enable desktop notifications

With this setting, some of the following options are disabled.

☐ Notify me about replies to threads I'm following

☐ Use different settings for my mobile devices

My keywords

Show a badge (1) in my channel list when a teammate uses one

e.g. sales, leads, lunch

Use commas to separate each keyword. Keywords are not case sensitive.

Do not disturb

When Do Not Disturb is turned on, Slack won't send you any noti
can override this if necessary.

☑ Automatically disable notifications from:

10:00 PM to 8:00 AM Pacific Daylight Time

Add a comment...

Submit

Kim: Is there a way to get notifications for new Slack threads too? 11/04/2017 6:11 PM

Aman: Yes, there's a setting for that now! I've indicated it in red. 12/04/2017 10:25 AM

What is this note about?

To turn on desktop notifications, open your Preferences to the Notifications tab, then click the green "Enable desktop notifications" button.

What task does this note relate to? (*)

notifications × preferences ×

Add a tag

273 8 1

☐ Keep it private?
☐ Needs an answer?

Who is this note best for?

First-time users (20%) [✓ Novice users (72%)] Expert users (8%)

Application: slack.com

(2 comments)

Last updated by Oliver
Feb 17, 2017 at 2:46pm

Save Delete Close

Figure 4: The curation interface used to view and edit notes and tutorial steps. This interface lets users (1) capture screenshots of the application or external help resources, (2) attach metadata such as textual explanations and task tags, (3) share notes via email, (4) participate in community discussions on each note, (5) mark notes as private or in need of an answer, (6) specify a suitable expertise level, (7) annotate the screenshot with visual markers, (8) view recommended similar notes or next steps (if available), and (9) minimize the current view (without exiting). The censor tool (10) can be used to permanently black out any sensitive information.

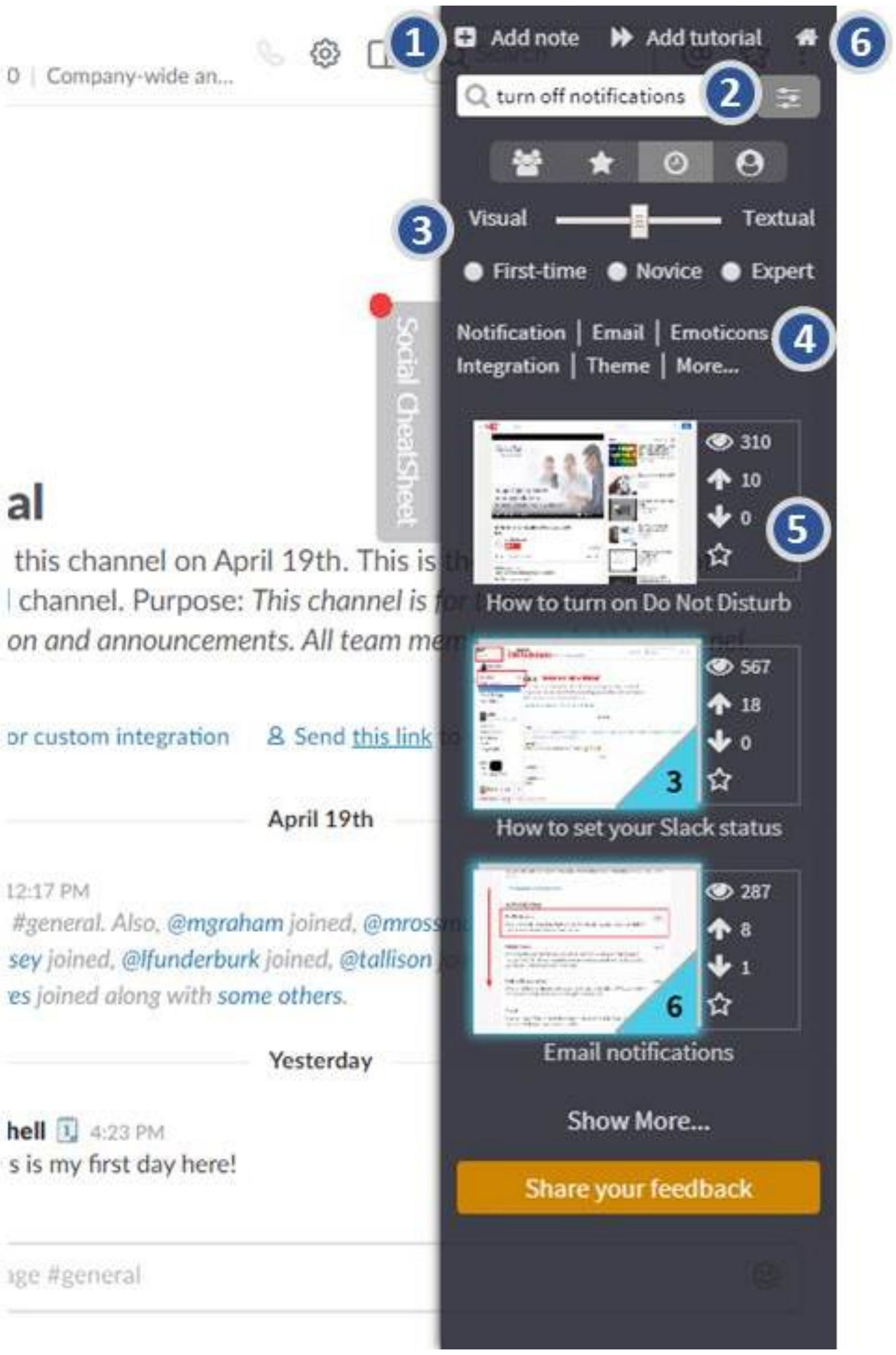


Figure 5: The Social CheatSheet drawer interface lets users (1) quickly add a new note or tutorial, as well as view a small selection of notes and tutorials below, which can be updated by (2) entering a natural language search query, (3) setting a learning style preference and expertise level, or (4) filter down to a predefined task tag. They can also (5) vote and favourite the retrieved notes, or (6) open the full overlay to see additional content.

Social CheatSheet also estimates the amount of text content in the captured region of the page. This information is used later on in determining the dominant learning style that the note exhibits. The new curation interface includes controls for participating in community votes on the note's helpfulness and the expertise level of the audience best suited to the note (Figure 4.6 and Figure 7). Furthermore, it includes the option to mark a note as private to oneself or flag it as a question needing an answer or clarification from the community (Figure 4.5 and Figure 7).

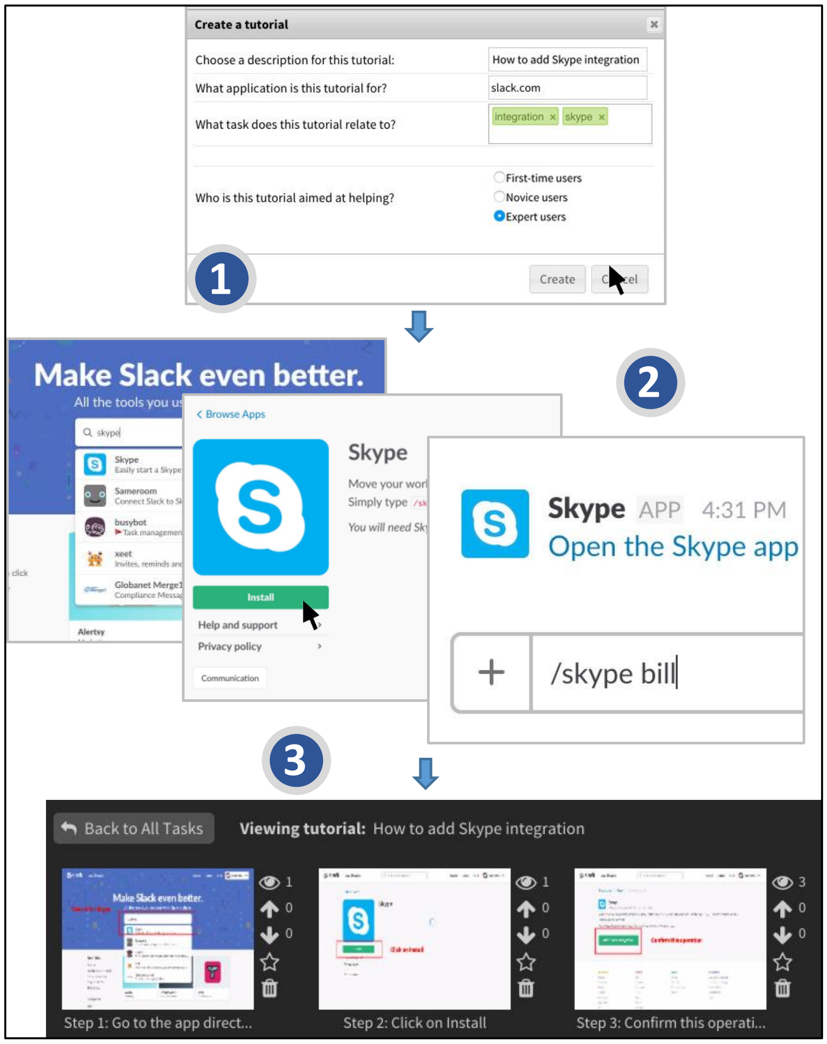
The Application field is automatically populated with the current domain name (Figure 4.2), to discourage the confusion that could result from multiple users choosing different names for the same application. Finally, once a note has been saved, a record of the user who last updated it and the edit timestamp are shown at the bottom.

For the convenience of users who prefer video tutorials, if the note was saved on a YouTube video page, Social CheatSheet recognizes this and allows future readers of the note to easily view an embedded version of that video from within the overlay.

Adding Tutorials

The original CheatSheet's automatic templating feature is a promising way to simplify the process of creating notes for multi-step tasks, but still results in a loose assortment of many individual notes without any means for a user to see what comes next in the sequence. To address the need for better organization and flow of help content during longer step-by-step tasks, Social CheatSheet adds the concept of "tutorials", a separate entity containing a sequential collection of steps each represented by a note.

The Create a Tutorial (Figure 3.1 and Figure 5.1) feature builds on CheatSheet's automatic templating, and is a special case of adding and grouping multiple notes together. An ordered sequence of unedited notes is captured automatically in the background while a task is performed, and these steps are added to a new tutorial object (Figure 3.8) to keep them isolated from the rest of the standalone notes. A key point here is that users can additionally attach helpful tips and snippets from other websites during the tutorial



Create a tutorial

Choose a description for this tutorial: How to add Skype integration

What application is this tutorial for? slack.com

What task does this tutorial relate to? integration × skype ×

Who is this tutorial aimed at helping? ○ First-time users ○ Novice users [✓ Expert users]

Viewing tutorial: How to add Skype integration

Step 1: Go to the app direct...

Step 2: Click on Install

Step 3: Confirm this operati...

Figure 6: The steps to create a tutorial. (1) First, metadata such as the title, tags, and difficulty level of the tutorial are specified. (2) Next, the user can perform their task normally, including visiting any external help sites, while Social CheatSheet captures screenshots of their actions in the background. (3) Finally, they can end the tutorial session by clicking on the browser extension's toolbar icon, after which Social CheatSheet displays the captured steps in the overlay, to be edited and curated as needed.

creation process (i.e. they are not limited to the target application itself). As before, users can curate their newly created tutorials after finishing, by adding annotations and metadata to individual steps and deleting any unwanted or redundant steps.

Furthermore, Social CheatSheet allows users to build tutorials from existing standalone notes by dragging them together in the grid interface. In this way, existing tutorials can also be extended and modified after the fact by adding additional steps, and individual notes can be packaged up and reused as an element of longer, more advanced step-by-step tasks without needing to be recreated from scratch.

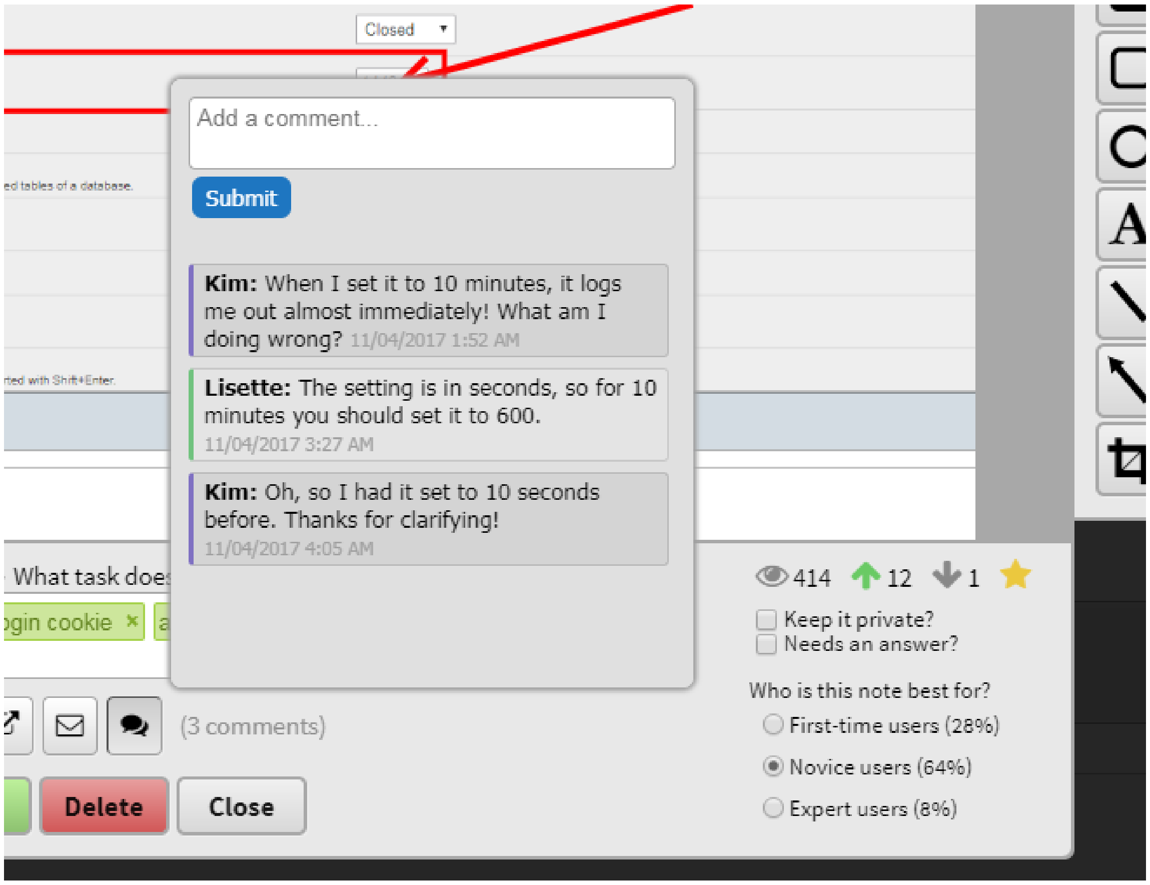
As an example, consider Bob, a manager, who has set up his team named Alpha on Slack.com, an instant messaging platform for team communication and collaboration. He wants to quickly explain how to integrate Skype with Slack to his team. He clicks on the "Create a Tutorial" feature, which brings up a dialog box (Figure 6) where Bob can specify the relevant metadata for his tutorial. Using his own annotated notes and extracted snippets from the official Slack documentation, Bob can easily create a multi-step tutorial that will be included in the Social CheatSheet help library for Slack. Additionally, Bob can use the Share by Email feature (Figure 4.3) to send this tutorial to team members who are currently not using Social CheatSheet (they will receive a link to a web page containing the tutorial).

Asking Questions or Seeking Clarifications

Based on the formative studies' finding that more community-based features were needed, Social CheatSheet includes several ways for users to interact with each other to seek and provide further help. When viewing an existing note or tutorial, a Social CheatSheet user can add comments (Figure 4.4 and Figure 7) to ask for clarifications or additional answers from the authors or the community. This also provides a simple mechanism for users to discuss changes to the note itself and improve the content together. It is also possible to mark a note as "needing an answer" (Figure 4.5 and Figure 7), which will flag it in the interface, ideally encouraging other users to provide edits or comments to resolve the issue. When marking a note as such, users can optionally specify an email address to subscribe to any activity on the note.

Preserving Privacy of Curated Content

Although the goal of our system is to be a community-curated platform for sharing help publicly, our formative studies highlighted the need to ensure user privacy is maintained throughout this process. Social CheatSheet allows a user to mark a note as private (Figure



Kim: When I set it to 10 minutes, it logs me out almost immediately! What am I doing wrong? 11/04/2017 1:52 AM

Lisette: The setting is in seconds, so for 10 minutes you should set it to 600. 11/04/2017 3:27 AM

Kim: Oh, so I had it set to 10 seconds before. Thanks for clarifying! 11/04/2017 4:05 AM

☐ Keep it private?
☐ Needs an answer?

Who is this note best for?
First-time users (28%)
[✓ Novice users (64%)]
Expert users (8%)

Figure 7: Social CheatSheet's new editor tools include the comment box, voting controls, expertise level controls, and flags for privacy or unanswered questions.

4.5 and Figure 7) to hide it from other users entirely if they wish. Alternatively, they can use the censoring tool to easily hide private names or other confidential information that might be present in the screenshot before saving the note publicly.

Finding Community-Curated Content

Once users have the Social CheatSheet extension installed, the in-application drawer (Figure 5) automatically retrieves the public notes and multi-step tutorials (marked by a blue symbol) for that website. This is in contrast the original CheatSheet [50] system which only allowed users to see their own notes. To ease the discovery of existing content and facilitate filtering, the drawer contains a list of the most popular tasks (Figure 5.4) within the current application, sorted in descending order by the number of notes and tutorials tagged with the task.

For example, consider Alice who just joined Bob's team Alpha through Slack's web interface and needs to quickly figure out its key features so she can keep up with her team. When she visits her Slack team page, Social CheatSheet immediately shows the 5 most popular notes on Slack in the drawer along with the top tasks (e.g., notifications, email, themes, mute, etc.). She clicks on the notifications tag and sees the most popular notes and tutorials related to that topic. She would be interested later in learning about muting notifications, so she bookmarks the tutorial by clicking on the star symbol (Figure 5.5).

When Alice clicks on "Show More", the drawer expands into the grid interface and shows a full list of tasks and content for the current application (Figure 3), appearing as an overlay across the page. When a user clicks on a task tag, the system filters down notes and tutorials to only show those tagged with the selected task.

Filtering and Sorting Content

Both the drawer and grid interfaces offer features to sort and filter content. By default, the most popular content is displayed first (based on views and votes). This can be changed to show the most recent notes, personal bookmarks, or only notes created by the user (Figure 3.4), or to filter the content by type such as tutorials or unanswered questions (Figure 3.5).

There are additional options to sort by expertise and by preference for visual or textual explanations (Figure 3.6 and Figure 8). For example, since Alice is a first time user of Slack, she clicks on the "First-time User" radio button to see only notes and tutorials appropriate for new users. Since Alice has found she learns best with more visuals and fewer text descriptions, she moves the learning style control towards showing more visual curated content.

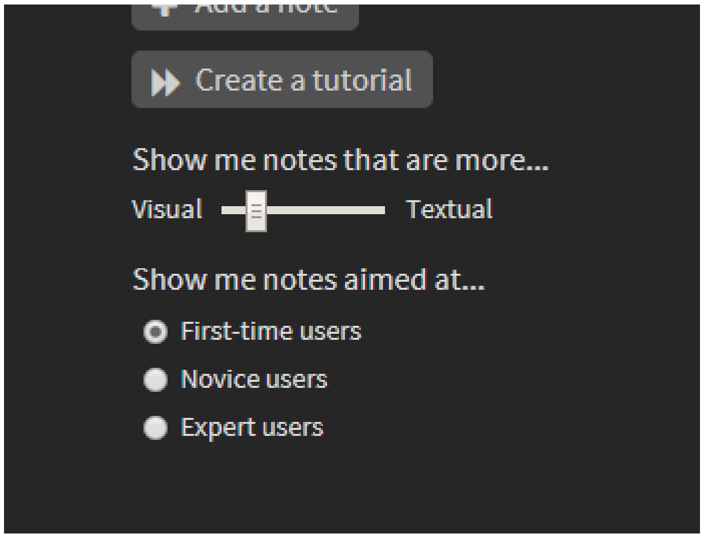


Figure 8: Controls to sort the library's retrieved notes by learning style preference and expertise level.

Searching for Content

Social CheatSheet places search bars in both the collapsible drawer (Figure 5) and the full overlay (Figure 3), allowing the list of notes to be filtered by a natural language query. The search algorithm (discussed in greater detail in Chapter 5) returns a ranked list of notes matching any portion of a note's description text, task tags, or textual annotations drawn over the screenshot, and these results are then used to update the list of notes shown in the display.

Viewing Content as Notes or Tutorials

When a user clicks on a note from the grid or the drawer, an expanded version of the screenshot opens up in the editor interface (Figure 4) along with annotations, descriptions, tags, other users' comments (Figure 4.4 and Figure 7), and other metadata (Figure 4.2). If the origin of the note was a YouTube video, users can immediately play the video within the editor. When viewing a standalone note, the system also makes recommendations for other similar notes along the left sidebar (Figure 4.8 and Figure 9). When viewing a tutorial step, this sidebar instead shows a list of the other steps within the same tutorial. Users can vote on the level of expertise that is commensurate with the note and also whether or not the note was helpful (Figure 4.5). Users can also click on the minimize button (Figure 4.9) to push the overlay down to the bottom of the screen, to easily go back-and-forth between their task within the application and the instructions on the note.

When a user clicks on a tutorial (marked by a blue symbol, Figure 3.8), all of the steps (individual notes) within the tutorial open up in the grid view (Figure 3). Clicking on a particular step opens up the editor (as described above), but instead of showing the recommended similar notes, the left side bar shows a quick shortcut to other steps in the tutorial.

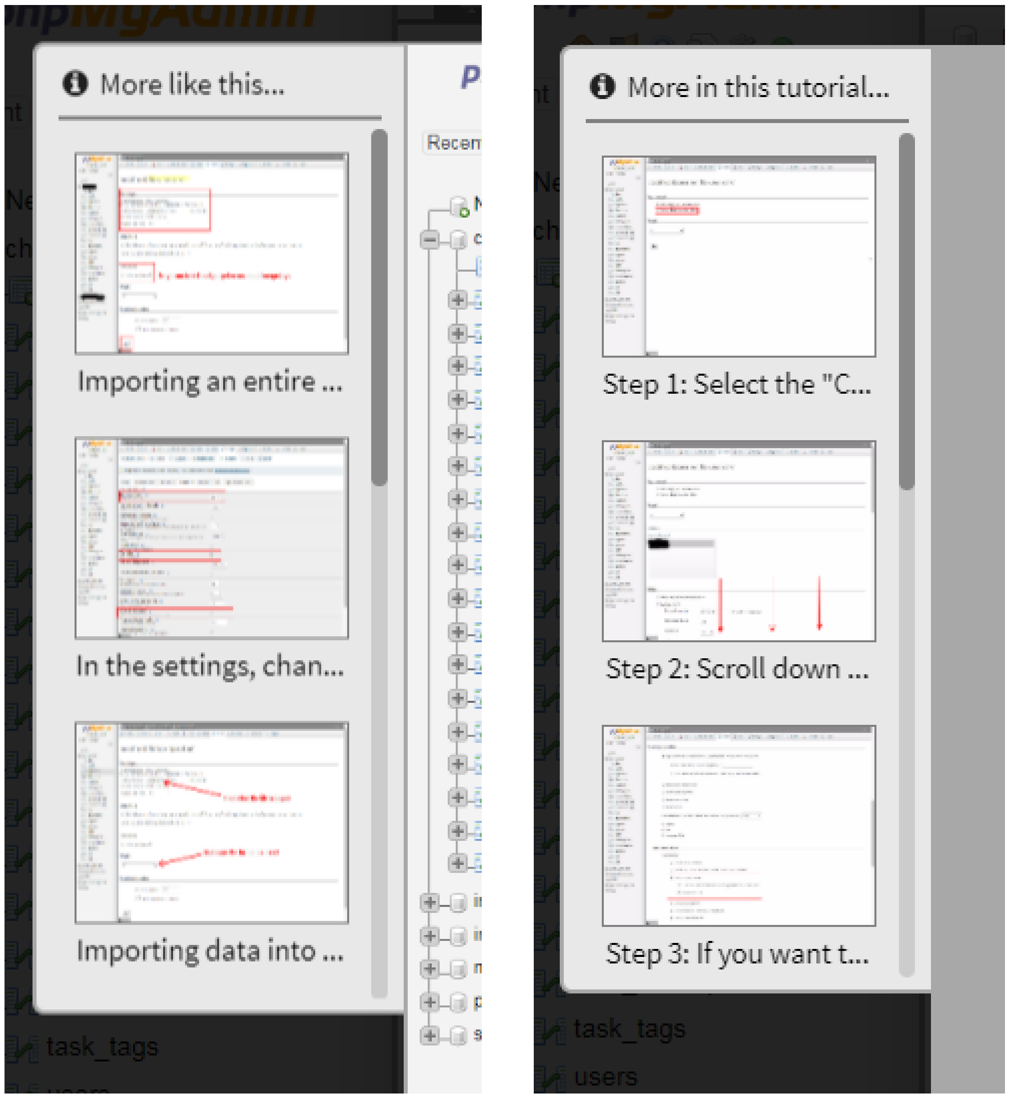


Figure 9: The sidebar on the left side of the note editor displays either recommended similar notes (on a standalone note, left), or other steps in the same tutorial (for a tutorial step, right).

Chapter 5
System Design and Implementation

To enable Social CheatSheet to function across any web application, we implemented it as a Google Chrome extension that automatically inserts its interface and content into each web page the user visits. Like its predecessor, Social CheatSheet works entirely independently of the underlying web application, and does not require any integration from web developers for it to run on their websites. For an end user, the only necessary setup is to install the browser extension from the Chrome Web Store.

The notes, tutorials, and related metadata are all stored in a database on a third-party server, with the use of a browser extension allowing Social CheatSheet data to be transferred to and from this server without violating the browser's same-origin policies on application websites.

Natural Language Search Algorithm

Search queries in Social CheatSheet are converted into search results using MySQL v5.7's full text search algorithm [42] with default parameters. This built-in algorithm uses a vector space model to assign each note or tutorial a relevance score based on its textual content, excluding common words from a preset stopword list. For our searches, the textual content of a note is the space-separated concatenation of is description, text annotations, task tags, and application name. The textual content of a tutorial is simply the space-separated concatenation of all of its steps' textual content.

Ranked Retrieval Approach for Showing Content

When a user of CheatSheet opens the drawer or the full overlay, or changes any of the search options within them, a retrieval request is sent to the Social CheatSheet server. This request includes the type of retrieval (*Most Popular*, *Most Recent*, *Favourites*, or *My Notes and Tutorials*), the name of the application for which they are requesting content, and the title of the page they are on. Optionally, it may also include any task tag or search query they have entered, a learning style preference, and an expertise level, if the user has modified any of these.

For a *Most Popular* retrieval request, the corpus of notes and tutorials for the given application is first optionally filtered to those having the specified task tag or matching a keyword in the search query, if applicable. Each note is then assigned a popularity score equal to [*upvotes* – *downvotes*/2 + sqrt(*views*)/10] (see Figure 10).

If a learning style or expertise level was specified, each note's score is scaled according to how closely its own calculated learning style and/or user-voted expertise level match the specified ones. The score is doubled for a setting that matches exactly, remains unchanged if a setting is as distant as possible, and scales linearly between these two extremes for all other cases. If the user also specified a search query, then the score for each note is multiplied by the search relevance score output by MySQL's full text search algorithm.

Scores for tutorials are calculated the same way, with their learning style and expertise level taken as the average of those values over all their individual steps. Finally, any notes (or tutorials containing notes) whose page title matches the specified page title are always ranked strictly higher than any without this property. The results are then returned in decreasing order of score.

```
Note score algorithm for "Most popular" retrieval
Inputs (* denotes optional user input):
  - The upvotes, downvotes, and views of the note, all non-negative integers
  - The note's learning style (either 0 or 1): note_lstyle
  - The user's learning style preference (a float between 0 and 1): user_lstyle*
  - The note's expertise level (a float between 0 and 2): note_exp_level
  - The user's specified expertise level (0, 1, or 2): user_exp_level*
  - MySQL's search query relevance score for this note: query_relevance*
Outputs:
  - The score for this note
Pseudocode:
  score = upvotes – downvotes/2 + sqrt(views)/10
  if user_lstyle is specified:
      score *= 2 – abs(user_lstyle – note_lstyle)
  if user_exp_level is specified:
      score *= 2 – abs(user_exp_level – note_exp_level)/2
  if query_relevance is specified:
      score *= query_relevance
  return score
```

Figure 10: Pseudocode for the popularity score algorithm

For *Most Recent* retrieval requests, the same filtering by application, task, and search query takes place, but then the notes are simply returned in reverse chronological order. A *Favourites* request returns all of the user's favourited notes and tutorials in the order they were favourited, and a *My Notes and Tutorials* request returns all of the notes and tutorials created by that user in the order they were created. Both *Favourites* and *My Notes and Tutorials* retrieve all relevant content regardless of application.

Approach for Recommending Similar Notes

To generate recommendations of other notes a user may benefit from viewing, Social CheatSheet keeps track of certain measures of similarity between pairs of notes. The similarity of any two notes is based on three more specific similarity scores: image similarity, text similarity, and task tag similarity.

The image similarity of a pair of notes is equal to the Hamming distance between the "perceptual hashes" of their annotated screenshots. For these perceptual hashes, our implementation relies on a simple 64-bit "difference hash" [28], which works by collapsing an image to a much smaller representation and then computing the differences in brightness between adjacent pixels. This perceptual hash has the property that visually similar pairs of images should often have hashes differing in only a few bits, while a visually distinct pair of images should typically result in two very different hashes. The difference hash algorithm could conceivably be replaced with any other similarly-accurate perceptual hash, but this one was chosen for its speed and simplicity.

Text and task tag similarity are both obtained by calculating pairwise TF-IDF scores, using the built-in TfidfVectorizer class in the scikit-learn package [61] to derive the document-term matrix M from the text of the two notes. The textual similarity score is then equal to the minimum entry of MM^T. A note's text, for this purpose, is the concatenation of its description and any textual annotations, separated by spaces. All of its task tags are similarly concatenated when finding the tag similarity score.

In the final similarity ranking, image similarity was given half the weight of text and tags in determining similarity, because in some of our testing we found that the images were less reliable for obtaining truly similar relevant results than user-curated text and tag similarity. Relying heavily on the annotated screenshots to determine similarity appeared to produce too many false positives due to similar-looking pages with different purposes. In contrast, the textual descriptions and tags were likely to contain keywords specific to the task at hand, making it easier to distinguish between unrelated notes.

Whenever a note is saved, its similarity to other notes in the system is recalculated based on the above components. Later, when a user requests to view a note in the editor, all of the other notes from the same application are ranked in descending order of similarity based on the pre-calculated values, and the top 10 results are returned as recommended similar notes.

Approach for Filtering by Learning Style Preferences

A note's dominant learning style is labeled as either Textual or Visual, calculated from a decision tree whenever it is saved. This decision tree takes as input an array of the following features:

- The combined length (in characters) of the note's description and text annotations
- The count of each type of annotation present on the note (8 features, one for each of pencil, highlighter, censor, line, arrow, rectangle, ellipse, and text annotations).
- An estimate of the fraction of the screenshot that consists of text (see Image Text-Fraction Estimation subsection below)

The tree is constructed using scikit-learn's DecisionTreeClassifier class [62] with default parameters and trained on a set of 48 hand-generated feature arrays, each hand-labeled as Visual or Textual by a researcher's inspection. The hand-generated feature arrays were designed to represent a realistic variety of notes emphasizing visual content, textual content, or a mix of both. In the decision tree, longer descriptions, higher text-fractions, and more highlighter annotations tend to branch toward a Textual label, while greater numbers of other annotations (e.g. arrows, rectangles) tend to branch toward a Visual label. In the future, the validity of this decision tree could be improved by training on a larger dataset derived from real notes created by Social CheatSheet users, and labelled according to the consensus of multiple users.

A note's dominant learning style is then used in the ranked retrieval algorithm (described above) in a comparison against a user-specified learning style preference, to allow users to retrieve content more suitable for the way they learn.

Image Text-Fraction Estimation

We initially implemented the text estimation using the Tesseract [54] optical character recognition library to analyze the screenshots for regions of text when they were saved on the Social CheatSheet server. However, we found that this method took too long on our hardware to process each image (at least 20 seconds for a single 1280x720 pixel screenshot). Moreover, the results were of highly variable accuracy, with the recognizer often perceiving text regions where there were none or failing to fully recognize real ones.

Given these difficulties and the unpredictability of screenshots taken from any page on the web, we instead opted to perform the text fraction analysis at the time the screenshot was captured, to leverage the additional information within the HTML Document Object Model (DOM) structure of the web page. Our custom algorithm filters the DOM text nodes on the page to find only those that are visible and currently inside the viewport when the screenshot is captured. It then divides the sum of their areas (in pixels) by the total area of the page to obtain the text fraction of the screenshot. This ignores any text that is part of an image (and hence not in a DOM text node), so on average it slightly underestimates the actual amount of text on the page. However, we attempted to account for this by manually tuning our learning-style decision tree, such that each branching rule that compares against a note's text-fraction is ~5% more likely to branch toward the side that favours a Textual label than it would have been in our unchanged tree.

This algorithm is linear in the number of DOM text nodes on the page, so it performs well for most practical purposes, although on extremely long web pages (upwards of 10,000 text nodes) it can result in a noticeable delay when each screenshot is captured. Future improvements could instead heuristically decide whether to run the full text node filter or simply rely on a cruder but faster estimate.

Chapter 6
Field Deployment of Social CheatSheet

How to best evaluate Social CheatSheet at this stage required consideration. We recognized that if we were to do an unconstrained "in the wild" field evaluation [1,8], it would take some time for independent help communities to organically emerge, and this process could be negatively impacted if there were significant usability issues (for which we had not yet tested). On the other hand, conducting another lab study (such as our formative studies) would not be as compelling as we desired, given the artificial setting. We settled on a hybrid approach similar in nature to Kleek et al.'s method [49], whereby we conducted a field deployment using a task-based approach, during which users were encouraged to explore the key functionality for curating and consuming content in the context of their own work. This was carried out for a period of one week.

Our goal with the task-based field deployment was to investigate the following research questions:

1. How do users perceive the usability and usefulness of Social CheatSheet?
2. How well does Social CheatSheet support our design requirements identified in the formative studies?
3. What factors would affect the long-term usage of Social CheatSheet, beyond the deployment period?

We asked participants to use Social CheatSheet both as curators and consumers on the Canvas web application, a learning management system in widespread use at Simon Fraser University (where the study took place). Canvas is a feature-rich application used by instructors to customize course content and by students to download lectures, submit assignments and view grades, among other more advanced tasks.

Participants

After obtaining ethics clearance, we recruited 15 study participants (7F, 8M) who were all between the ages of 19-34 and currently studying at Simon Fraser University (6 Bachelors, 6 Masters, 3 PhD) in various majors. We posted flyers on university bulletin boards, and university mailing lists to recruit participants who were currently using Canvas for a course either as a student or as a teaching assistant (TA). Most of the participants were frequent users of Canvas: seven said they used it daily, five said they used it at least once a week, and the remaining three said they used it once a month or less. None of the participants knew each other and we gave them anonymous user IDs during the deployment.

Deployment Protocol

Our week-long deployment was carried out in three stages, yielding different data points throughout the study.

Pre-deployment Setup: Study participants first attended a brief 20-30 minute setup session, during which they filled out a demographic questionnaire, and set up the extension on their personal laptops for the deployment period. Participants were then given a brief demonstration of how to access Social CheatSheet in the browser, an overview of the features available to them, a handout listing the example tasks, and another brief handout explaining the list of Social CheatSheet features just described to them. Prior to leaving, participants were remunerated with a $20 gift card and were asked to sign up for a post-deployment interview time slot.

Deployment Period: We asked participants to use Social CheatSheet on their own time, as they saw fit, with their interactions in Social CheatSheet being logged to our server. We provided them with 6 example tasks that differed slightly depending on whether the participant was a TA or a student (due to different capabilities in Canvas). For the basic tasks, we asked participants to assume they were curating help content to explain the tasks to new students or new TAs. For example, we asked students to create content about forming project groups and submitting assignments, and TAs to create content about grading assignments and creating new discussion threads. For the advanced tasks, such as using the real-time video streaming and the Calendar Scheduler, we asked participants to try to learn these features and create curated content for their own reuse.

As part of the bootstrapping process, three researchers pre-populated Social CheatSheet with a dozen notes and tutorials based on Canvas FAQs available on our university website (these did not overlap with the tasks provided to participants).

We sent daily emails to remind participants to try Social CheatSheet, and to fill out an open-ended online feedback form to identify any usability issues they encountered and/or any thoughts on using the tool. The activity logs saved detailed timestamped information about user activity within Social CheatSheet (based on clicks on any interactive feature and opening or closing of the overlay). No other personally identifiable information or page URLs were logged.

Post-deployment Interviews: Study participants returned for a post-deployment interview and filled out an exit questionnaire about their experience with our system. Our semi-structured interviews with participants lasted about 30 minutes and probed into user perceptions of Social CheatSheet based on their usage over the past week (often by showing the researchers some of the notes or tutorials they had created), how they felt about themselves or others using it in the future, and how it differed from ways they might normally give or receive help. Finally, the researchers helped participants uninstall the extension, and provided them with an additional $30 gift card for completing the deployment and the interview.

Our post-deployment questionnaire included five questions about perceptions of Social CheatSheet. All of these questions were measured on a 7-point Likert scale.

1. How would you describe the overall ease of use of Social CheatSheet?
   (Very difficult to use — Very easy to use)
2. As a potential learning aid for new users, Social CheatSheet can be…
   (Very unhelpful — Very helpful)
3. As a potential learning aid for expert users, Social CheatSheet can be…
   (Very unhelpful — Very helpful)
4. Given the option, I would continue using Social CheatSheet on Canvas or other websites. (Strongly disagree — Strongly agree)
5. I would recommend the Social CheatSheet system to other users.
   (Strongly disagree — Strongly agree)

Data Analysis

As described above, we used multiple sources to collect data about participants' usage of Social CheatSheet and their perceptions via pre/post-deployment questionnaires, usage logs, online feedback forms, and interviews. We used an inductive analysis approach [48] to look for recurrent themes and triangulated results from our data sources.

Overview of Usage

In the suggested curation tasks, we had given participants the freedom to try creating standalone notes or tutorials. But, as shown in Table 1, tutorials appeared to be more popular among participants than notes. Likewise, in terms of consuming content created by others, participants also appeared to prefer using tutorials over notes. It was interesting to see that many participants tried different features on their own (even though they were not suggested in the task). For example, two participants tried using the commenting features: one tried answering another participant's question on a note that was marked as needing an answer; another participant added a clarification on her own tutorial step. Since the participants in the study were anonymous, and none indicated that they knew any of the other participants personally, it is encouraging to see some evidence of direct social interactions during the deployment.

One of our most encouraging findings was that two-thirds (10/15) of our participants

Usage Sessions	
Avg # of usage sessions per user	9.1
Avg length of session (min) per user	9.5
Avg daily time spent (min) per user	12.7
Avg # of online surveys submitted per user	2.9
New Content Added during Deployment	
Total standalone notes created	20
Total tutorials created	80
Avg # of notes and/or tutorials created per user	6.7
Features Used during Deployment	
Questions asked	7
Notes marked private	6
Shares by email	4
Comments posted	4
Expertise level changes	14
Learning style changes	35
Upvotes and downvotes	62
Search queries issued	222
Task tags clicked	30
Censor tool uses	239
Note views	136
Tutorials views	156

Table 1: Summary of Social CheatSheet usage

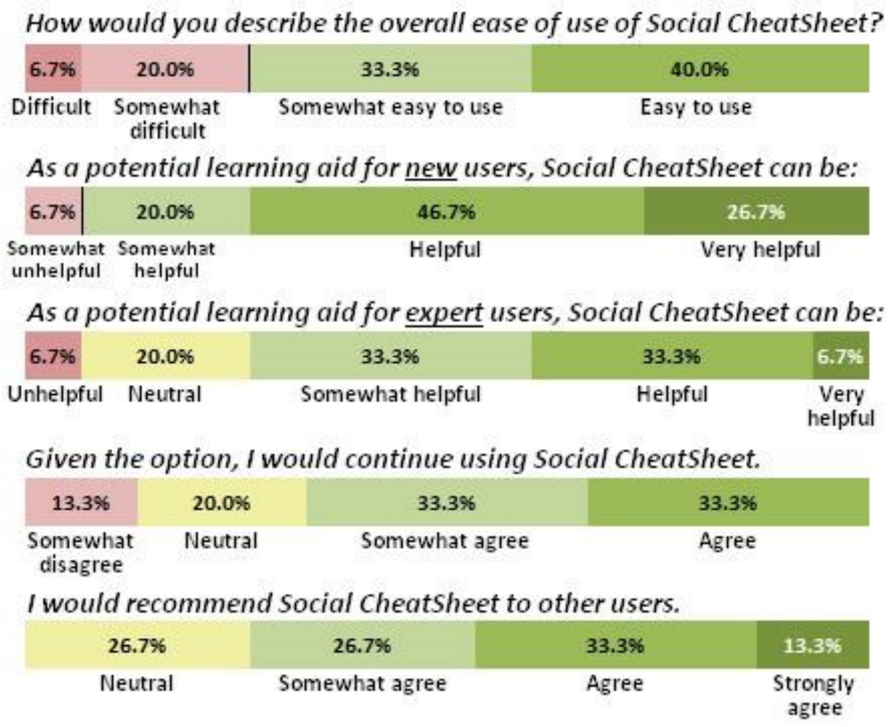


Figure 11: Post-deployment questionnaire results. All questions were on a 7-point Likert scale. Scale categories that received zero responses are not shown.

spontaneously used Social CheatSheet beyond the Canvas application, as evidenced by our log data. They added notes or tutorials on 12 other sites, such as our university registration site, a teaching evaluation tool, Amazon, and Google Sites, among others. This non-Canvas activity comprised 25% of the all notes created and 19% of all tutorials, which was surprising given that none of our example tasks directly asked participants to use these sites. We discuss the reasons for this phenomenon in a later section.

Usability and Usefulness of Social CheatSheet

The post-deployment questionnaires indicated that the majority of participants (11/15) found Social CheatSheet easy to use (Figure 11). Participants highlighted several reasons for this, such as the visual nature of the curated content:

> *"I guess the main advantage of this is that I can have something visual that replicates the actual screen, or the interface that they're going to be using… and also the ability to really highlight stuff or, you know, write comments on the screen itself, so that's something beneficial." (P02)*

All 15 participants mentioned that they found it useful to see other people's curated notes and tutorials. Interestingly, some participants who considered themselves to be frequent expert users of Canvas revealed that they still learned something new by browsing the content curated by other users:

> *"I saw people use Canvas in ways I'd never used it before, and that was interesting…and to see how people responded to the same task, like they did it in different ways, it was good… I was also able to learn a lot from just checking through the feedback from other people." (P04)*

Users who would otherwise turn to a collection of external tools and applications for saving and sending help found Social CheatSheet easy to use for streamlining both the creation process and the means of sharing with others:

> *"I usually use the photo editor…the one for painting on the computer… I use screenshots with some social media apps… But with this [Social CheatSheet], I don't have to switch between other applications: I just use Social CheatSheet and it takes the screenshot automatically… it's easy." (P13)*

In addition to being easy to use, participants overwhelmingly indicated that Social CheatSheet was useful in locating relevant curated content in different contexts. For example, they recognized the benefits of social curation compared to using static help, such as documentation created by application owners:

> *"Maybe sometimes they [in-app help creators] didn't think through a lot of usage scenarios, but in Social CheatSheet since many persons [can contribute], it's possible to find the information you hope to find." (P01)*

As shown in Figure 11, almost every participant (14/15) felt that new users to an application would find Social CheatSheet helpful. The common reason was that new users often face a "blank-slate problem" and do not even know what they need to look for or how to find it. The visual annotations and step-by-step nature of the content could be particularly beneficial for these users:

> *"I would say Social CheatSheet is more beneficial to new users and novice users more than the expert users… It does take a lot of screenshots, and it's very detailed in that way. So that kind of level of detail would benefit novice users more than an expert user who may prefer less numbers of slides with more content condensed into each particular slide." (P02)*

Although participants were less enthusiastic about the helpfulness for expert users compared to novices, a strong majority of them (11/15) still indicated that Social CheatSheet would be helpful for experts (Figure 11). These participants felt that expert users could use Social CheatSheet to share their knowledge with others, even if they would not necessarily find help for themselves:

> *"If I am an expert… I'm not going to use it for solving my questions. I may be willing to solve others' questions if I have time, if it's easy for me to use the tutorial thing." (P11)*

However, another participant put forth the distinction that novice users would use Social CheatSheet to accomplish basic, task-specific goals, while an expert might take a more holistic, long-term approach to learning new features:

> *"An expert user, when they browse for new features that they haven't learned about yet, then they'll be thinking about how to incorporate it into their usage of the app later on, but a novice user… probably just wants to get a basic 'how to do this, how to do that' understanding." (P14)*

As expected, given that this was the first evaluation of Social CheatSheet, participants did identify some usability challenges and bugs. We collected 44 such issues through the online feedback survey during the deployment. A recurring theme was that it was very easy to create tutorials, but it would be helpful to have more fine-grained control or feedback on the steps being recorded. Participants also provided a variety of suggestions for improving the user experience, such as including audio clips in notes, having advanced features to arrange and mass-edit tutorial steps, and allowing more customization of the annotation tools in the editor.

Social CheatSheet's Support for Design Requirements

We also had several key findings in relation to the five design requirements from the formative studies.

Community-oriented features: Overall, we found that users benefited from having community-oriented features to easily browse and locate useful content. For example, even expert users of Canvas indicated that they often forget certain features over time or did not know about certain advanced or hidden "power-user" features. Social CheatSheet was well-suited for discovering such features with community-generated tips and reminders:

> *"I was a TA [before], and I forgot some of the basic issues in Canvas…so, for the people who are experts for a period of time and then they don't use [an app] for a long time, [Social CheatSheet] can be a great tool to refresh their mind." (P01)*

Additionally, viewing other users' help content seemed to encourage some participants to experiment more on their own and use the application in new ways:

> *"I didn't know about most of those things that we can do in Canvas… so I learned about some aspects using other people's tutorials, and then tried to do the same with some modifications myself." (P12)*

It was interesting to see that many participants tried different community-oriented features on their own (even though they were not explicitly mentioned in the provided tasks). For example, two participants tried using the commenting features: one tried answering another participant's question on a note that was marked as needing an answer; another participant added a clarification on her own tutorial step. The email feature was used four times to send notes to other people who were not Social CheatSheet users and participants found it intuitive and useful for sharing content with their parents or friends. Several more indicated that the email feature can be useful in the future, such as to help older adults who may not want to install special extensions:

"Once [we] make tutorials with many steps, then, for example elderly people and some people who are not familiar…can just follow one step by step, and it's pretty clear to explain what should I do on this website." (P10)

Additionally, over half of the participants indicated that the ability to ask questions to the community was useful and easy to use, with 6 different participants actually creating at least one note marked as needing an answer. Since the participants in the study were anonymous, and none indicated that they knew any of the other participants, it is encouraging to see some evidence of direct social interactions during the deployment.

Step-by-step tutorials: As shown in Table 1, tutorials were much more popular than standalone notes in terms of usage and perceived utility, with all but one participant marking the tutorial feature as "useful" or "very useful". This was higher than standalone notes, for which only 11/15 participants said the same, and which one participant did not use at all. Consistent with our formative studies, we discovered that participants preferred to break apart the task into different steps rather than show everything on a single image:

"I really like the tutorial function...because you can see all the [notes] together and see the context of using some function" (P03)

Personalization options: We observed that just over half of the participants (8/15) tried changing the value of the learning style slider at some point during the deployment, and among these participants, all but one used it multiple times. Similarity, 7/15 participants tried filtering notes by expertise level, with all but one of these setting it to a value of "Novice", and the remaining one to "Expert". We were pleased to see that users were interested in trying these features without any prompting. While these personalization features were currently manually controlled by users, some users were curious to see how the system could learn their preferences automatically over time and show relevant content accordingly.

Preserving privacy: Although not explicitly stated in any task, four participants marked notes as private. This may not be indicative of realistic usage, since a primary motivation for that feature is to keep a reference for oneself in the future, and this use case is particularly ill-suited to a short-term deployment. While screenshots of Canvas were likely to contain private information such as student names or grades, most users (11/15) preferred using our censor tool to "black out" these sections of the screenshots instead (rather than making the note private). Several participants indicated that they would prefer to see the censor streamlined such that they could hide same parts of multiple similar tutorial steps, or specify areas to censor upfront when creating a tutorial.

Flexible natural language search: While it is not a research contribution, our improvements to Social CheatSheet's search engine did improve user experience. Participants in the field deployment appeared to use search queries as their primary means of navigating the help resources within Social CheatSheet, issuing 222 freeform search queries in total (7.4x more than their use of task tags). This differs substantially from the usage patterns during the formative lab study, where participants used 1.3x more task tags than search queries on average due to limitations of the original search algorithm.

Potential for Long-Term Usage of Social CheatSheet

One of our most encouraging findings was that two-thirds (10/15) of our participants spontaneously used Social CheatSheet beyond the Canvas application. Our logs showed that they added notes or tutorials on 12 other sites, such as our university registration site, a teaching evaluation tool, Amazon, and Google Sites, among others. This non-Canvas activity comprised 25% of the all notes created and 19% of all tutorials, which was surprising given that none of our example tasks directly asked participants to use these sites.

We were also encouraged to see the majority of participants (10/15) agreeing that they would continue to use Social CheatSheet after the deployment, if they had the option to do so (Figure 11). Furthermore, 11/15 indicated that they would recommend Social CheatSheet to others. In our interviews, participants enumerated several reasons why they would keep using Social CheatSheet — application independence and ease of use were cited as key reasons:

> *"I think the biggest advantage is it's a plugin in the website, to any website. Whenever you want to see how to use the website you can just click on CheatSheet and then do some records. (P03)*

Another participant explained that Social CheatSheet could help her avoid the social cost of calling someone and re-learning a complex process (e.g., for installing software):

> *"I think it [Social CheatSheet] would be useful for downloading and setting up software that requires following some instructions, because sometimes when you've done it before and you want to do it a second time on a different system, you can't remember and have to go through the whole process again." (P04)*

This use case also reflects the case where a user knows she will need help with an infrequently-performed activity in the future, and takes notes to provide help for her future self.

Another participant talked about using Social CheatSheet for coding-related use cases (as a curator and consumer):

> *"…if I were to try to explain a code snippet to someone...then I can just create a tutorial using this [Social CheatSheet] and send an email to them. So, I guess it just has to do with how it's relevant, the tutorial I'm making is, to my own needs." (P02)*

Some participants mentioned that they would only create notes and tutorials if they had a very specific need to do so, such as a family member asking for help with a software problem, but not simply to share their knowledge unasked. Overall, most participants indicated that they would be more likely to use Social CheatSheet as a content consumer than as a content creator, in part because of the additional work required to curate high-quality content. One participant who was not a frequent user of Canvas said that he did not see himself contributing content to Social CheatSheet about Canvas, but that for other applications that he was more familiar with and used regularly, he would add notes:

> *"The thing is...I don't use [Canvas] that frequently, so I don't make notes…but if it's some application [e.g., Google Search] that I'm using every day, I might add notes." (P15)*

While users were overall positive about accessing curated notes and tutorials, some participants felt that they did not find anything helpful due to the limited content currently available, but that the system could be more useful in the future:

> *"I found myself using it to try to find answers to the things I didn't know how to do in Canvas, but unfortunately there were no answers there yet. But I thought it was really handy how you can search things quickly and… if I was to do it next week then it would be probably more useful." (P08)*

Another person pointed out that even though he found relevant content and found the visuals to be helpful, the metadata and description quality could be improved:

> *"I searched through the tutorials to see if I could find if anyone else had submitted a tutorial for [a] scenario. I found some…but they were very sloppy, and I couldn't get any information from them." (P07)*

Other participants were less concerned about the content quality because they felt that it was just a matter of adding moderation features and quality checks that are already common on forums and other social applications.

Chapter 7
Discussion

Our design and evaluation of Social CheatSheet demonstrates how users could participate and benefit from community-curated help, particularly multi-step tutorials, atop feature-rich web applications. More broadly, our work offers insights into a how to design a useful collaboratively constructed information space [2] that not only leverages the benefits of the community, but also offers users a private space for their own content and ability to personalize retrieval. Although our evaluation was a short-term small-scale deployment, it was encouraging to see that users could still find useful curated content, benefit from trying out the system on other applications, and be willing to keep using the system beyond the deployment. Nonetheless, our findings also highlight some limitations that future work can address to improve different aspects of social curation in the context of learning software, as we outline below.

Incentivizing Social Curation and Community-Building

Our findings showed that a third of the users would likely not add their own curated notes and tutorials. This is not surprising given other findings on online communities where most users are consumers rather than contributors [3]. However, given the ease of using the curation features, over half of the users were neutral and not opposed to adding content if the community were to grow in the future. Also, since many users already curate content for helping family and friends or even take notes for themselves to save helpful instructions

[50], it may just be a matter of further incentivizing users to participate and adjust to the Social CheatSheet medium for curation. Furthermore, given the long-term growth of technical help communities such as Stack Overflow [32] and software discussion forums [30], we have some confidence that Social CheatSheet can eventually grow as a community portal.

The next step in this research is to use our platform to investigate different aspects of user participation via controlled field experiments. For example, does the ease of visual curation entice more people to participate than a traditional text-based curation interface? How can we employ techniques such as gamification [11,12] that have worked well in other domains to improve social curation? How would participation rates compare across different domain-specific communities (e.g., users of spreadsheets, 3D modelling software, or health applications)?

In our field study, users retrieved content added by other anonymous users and they did not express any major concern about not knowing a contributor's identity or reputation. However, as has been shown in previous work, content quality and user perceptions in the long run can be affected by contributions made by experts vs. non-experts [37]. It would be interesting to explicitly investigate how users perceive content added by anonymous users (as they did in our deployment study) versus users revealing their identities and additional information about their expertise.

Even though our primary goal was to design Social CheatSheet as a community-based help system that is independent of the application owners, there is opportunity for technical app support staff to also help their users through this platform. Support staff already contribute content on Twitter, Facebook, and product forums [53] — similarly, they could use Social CheatSheet to broadcast important instructions, changes, or tips to users that would appear within a user's interface (but no other technical integration work would be required on the part of application owners).

As user contributions grow and more content is added to the social curation platform, naturally a concern may be if the user experience would be compromised in some way. Future work could investigate how well the retrieval algorithms and the usability of the system scale with active user communities and growth in curated content.

Maintaining Quality and Availability of Curated Help

Despite the overall positive feedback about retrieving help, users did raise some concerns about the long-term availability of relevant high-quality instructions and related metadata. As we explore mechanisms for incentivizing users to participate in social curation, we believe it is important to also design mechanisms to ensure proper maintenance of community-curated content. In particular, given the rapid evolution of web applications, it is important to consider how the content and metadata could be maintained to preserve relevance. In our current model, the out-of-date information will get down-voted by the community, but future work could explore how to automatically detect web page changes and compute a "freshness" score of the related help content.

Even though Social CheatSheet is a community-based help system that does not depend on application owners, there is opportunity for technical app support staff to help their users by maintaining the availability and quality of the help content. Support staff already contribute content on *Twitter*, *Facebook*, and product forums [51], often by providing "verified" answers — similarly, they could use Social CheatSheet to broadcast important instructions, changes, or tips to users that would appear within a user interface.

Personalizing and Recommending Targeted Curated Help

Although users found it easy to locate and filter relevant curated help, there may be an opportunity to further push the boundaries of our social curation platform and offer more targeted curated help to users. For example, we can try to infer what tasks users are actually performing in the application and proactively recommend relevant curated help (this is in contrast to other innovations that focus on recommending commands and features [34,38]). Future work can also investigate how to build hybrid help systems that have knowledge about the characteristics of users participating in the curation platform and can use collaborative filtering approaches [43,44] to suggest more targeted content. This could lead to a far more sophisticated algorithm for determining which notes to recommend to a user, beyond the current simplistic approach of comparing only the visual and textual content of the notes themselves.

Furthermore the perceptual image hashes that are used as a measure of notes' image similarity are designed to perform accurately on natural images such as photographs, but not necessarily on interface screenshots. This is most likely a key contributor to their unsuitability as a note similarity metric in comparison to the text-based approaches we used. It may prove fruitful in the future to investigate whether Social CheatSheet could use alternative means of deducing the similarity of web interfaces via screenshots, such as pixel-based reverse engineering [13].

Feedback from both our formative and field studies indicated that some users may learn better from either visual or text-based instructions, and that novices and experts prefer different granularity and format of help content. Yet, most current software learning materials rarely take individual user characteristics into account and offer the same help to all users. While our system begins to explore the idea of personalizing curated help content by allowing users to retrieve different content based on their expertise (e.g., novice to expert) and learning style [19] preferences (e.g., visual or textual), there are many opportunities for more automatic and adaptive personalization methods. For example, it may be possible to extend and integrate HCI research on automatically detecting novice versus expert behaviours [20] and supporting novice to expert transitions [9]. Similarly, mechanisms for automatically inferring learning styles have already been explored in educational settings [19,27] and it would be interesting to investigate how they could be adapted in the context of using software and accessing help.

Lastly, there is potential to explore the benefits of social in-application help in contexts beyond troubleshooting or seeking help with feature-rich applications. For example, there is already evidence of online sharing ecosystems around customizations [21], but there is little in-application support for this type of sharing. Perhaps platforms such as Social CheatSheet can be extended to allow users to share such content. Personalization of socially curated content can also be investigated in educational contexts, especially the potential of adding curated content to large-scale distance learning platforms, such as MOOCs.

Challenges and Limitations of the Field Deployment

In running our field deployment of Social CheatSheet, we noted a few limitations and potential improvements to the protocol. First, because of the short duration and small population of participants in the study, there was little potential for a realistic community to develop within the Social CheatSheet environment. Although we did see some basic levels of social interaction from some of the participants, a longer and less constrained field study would be necessary to observe whether these interactions are likely to grow into sustained collaborations over time.

Furthermore, despite the fact that we provided deployment participants with tasks to complete in Canvas, most of them still independently tried creating notes and tutorials on other websites. Since some also noted that they would prefer to use Social CheatSheet on other applications that they use more often than Canvas, it may be beneficial for future deployment studies to provide more open-ended tasks that give participants greater flexibility in where and how they complete them, to improve the realism of the study and keep participants engaged.

System Design Challenges and Ideas

In the process of designing and implementing working prototypes for Social CheatSheet, there were a number of technical hurdles to overcome. First and foremost, a system that overlays a substantial amount of content atop any (unpredictable) web page must deal with the possibility that CSS style rules from the underlying page may interfere with the overlay and vice versa. In some of the initial iterations of our CheatSheet prototype, this resulted in inconsistencies in the layout and styling of elements within our overlay and drawer between different pages, which occasionally made the interface virtually unusable on certain host websites.

Ultimately, we chose to solve this problem using the *Cleanslate* [63] stylesheet, which reverts to their defaults most of the CSS rules set by the underlying page, solely within the scope of our Social CheatSheet elements. Unfortunately, this also requires that Social CheatSheet abandon many of the benefits of the style cascade by using the `!important` CSS directive on nearly every rule, which limits the maintainability of our stylesheets. Luckily, at the time of writing, there is a proposal [64] under consideration to add a `revert` keyword to the CSS specification, which would perform a similar function to *Cleanslate*, but without this key drawback. Alternatively, another solution that could be implemented in the meantime would involve displaying the entire Social CheatSheet interface within one or more HTML iframe elements, which would eliminate any unwanted stylesheet interactions at the potential cost of introducing issues involving cross-origin resource restrictions.

Another design element of Social CheatSheet which was not fully explored was the recognition of video content, to be later embedded and playable along with a note or tutorial. Our current implementation is quite limited, only working for YouTube videos, and without any information on which parts of the video are helpful for the task at hand. A number of participants, in both the formative studies and the field deployment, mentioned that they rely largely on video content when seeking software help online, so it could prove useful to many software learners to have Social CheatSheet integrate better with online video resources. A simple improvement might be to allow the creator of a note to specify start and end times for the captured video, so that future viewers would avoid the need to find and seek to the relevant portions. Better yet, Social CheatSheet could potentially try to infer the desired start and end times automatically, either based on attributes of the video element or from user behaviour on the page.

Finally, even if we were to have a perfectly functioning system implementation, we would still need to consider some of the social challenges that might arise as part of a community-based system like Social CheatSheet. Some of the deployment participants raised the issue of user and content moderation within Social CheatSheet, such as resolving edit disputes between users, dealing with malicious users who vandalize others' carefully curated content, and preventing voting or commenting abuse from "sock puppet" accounts. Although the current implementation is not suited to address all of the issues raised, future improvements could follow common solutions to these problems, as employed by other online communities. For example, these might involve imposing limitations on the degree to which new user accounts can edit existing content, or allowing the community to select trusted moderators with greater privileges.

Chapter 8
Conclusion

In conclusion, we have contributed a new vision for enabling social curation of software help resources from within any existing web application. The two formative studies that we ran provided initial design goals and user feedback leading to the development of the Social CheatSheet system. This system offers users the ability to access and contribute to a public visual cheat sheet containing task-specific tips, instructions, and tutorials, all from a client-side overlay independent of the underlying web application. Our field deployment of this new system further contributed insights into the benefits of using community-curated software help resources and the behaviour of users when first given access to such a tool. Finally, this study contributed implications for better incentivizing social curation and community-building in future similar systems, and recommending personalized curated help content to software learners where and when they require it.

References

1. Mark S. Ackerman. 1998. Augmenting Organizational Memory: A Field Study of Answer Garden. *ACM Trans. Inf. Syst.* 16, 3: 203–224. https://doi.org/10.1145/290159.290160

2. Mark S. Ackerman, Juri Dachtera, Volkmar Pipek, and Volker Wulf. 2013. Sharing Knowledge and Expertise: The CSCW View of Knowledge Management. *Comput. Supported Coop. Work* 22, 4–6: 531–573. https://doi.org/10.1007/s10606-013-9192-8

3. Charles Arthur. 2006. What is the 1% rule? *The Guardian*. Retrieved from https://www.theguardian.com/technology/2006/jul/20/guardianweeklytechnologysection2

4. Andrea Bunt, Patrick Dubois, Ben Lafreniere, Michael A. Terry, and David T. Cormack. 2014. TaggedComments: Promoting and Integrating User Comments in Online Application Tutorials. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems* (CHI '14), 4037–4046. https://doi.org/10.1145/2556288.2557118

5. John M. Carroll, Penny L. Smith-Kerker, James R. Ford, and Sandra A. Mazur-Rimetz. 1987. The Minimal Manual. *Hum.-Comput. Interact.* 3, 2: 123–153. https://doi.org/10.1207/s15327051hci0302_2

6. Parmit K. Chilana, Andrew J. Ko, and Jacob Wobbrock. 2015. From User-Centered to Adoption-Centered Design: A Case Study of an HCI Research Innovation Becoming a Product. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI '15), 1749–1758. https://doi.org/10.1145/2702123.2702412

7. Parmit K. Chilana, Andrew J. Ko, and Jacob O. Wobbrock. 2012. LemonAid: Selection-based Crowdsourced Contextual Help for Web Applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '12), 1549–1558. https://doi.org/10.1145/2207676.2208620

8. Parmit K. Chilana, Andrew J. Ko, Jacob O. Wobbrock, and Tovi Grossman. 2013. A Multi-site Field Study of Crowdsourced Contextual Help: Usage and Perspectives of End Users

and Software Teams. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '13), 217–226. https://doi.org/10.1145/2470654.2470685

9. Andy Cockburn, Carl Gutwin, Joey Scarr, and Sylvain Malacria. 2014. Supporting Novice to Expert Transitions in User Interfaces. *ACM Comput. Surv.* 47, 2: 31:1–31:36. https://doi.org/10.1145/2659796

10. Allen Cypher, Mira Dontcheva, Tessa Lau, and Jeffrey Nichols. 2010. *No Code Required: Giving Users Tools to Transform the Web*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

11. Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. 2011. From Game Design Elements to Gamefulness: Defining "Gamification." In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments* (MindTrek '11), 9–15. https://doi.org/10.1145/2181037.2181040

12. Sebastian Deterding, Miguel Sicart, Lennart Nacke, Kenton O'Hara, and Dan Dixon. 2011. Gamification. Using Game-design Elements in Non-gaming Contexts. In *CHI '11 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '11), 2425–2428. https://doi.org/10.1145/1979742.1979575

13. Morgan Dixon, Daniel Leventhal, and James Fogarty. 2011. Content and Hierarchy in Pixel-based Methods for Reverse Engineering Interface Structure. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '11), 969–978. https://doi.org/10.1145/1978942.1979086

14. 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), 115–124. https://doi.org/10.1145/1166253.1166273

15. Dan Whaley. Home. *Hypothesis*. Retrieved July 25, 2017 from https://web.hypothes.is/

16. Evernote Corporation. Get organized. Work smarter. Remember everything. *Evernote*. Retrieved July 22, 2017 from https://en-prod-wwwsite.appspot.com/

17. Adam Fourney, Ben Lafreniere, Parmit Chilana, and Michael Terry. 2014. InterTwine: Creating Interapplication Information Scent to Support Coordinated Use of Software. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology* (UIST '14), 429–438. https://doi.org/10.1145/2642918.2647420

18. C. Ailie Fraser, Mira Dontcheva, Holger Winnemoeller, and Scott Klemmer. 2016. DiscoverySpace: Crowdsourced Suggestions Onboard Novices in Complex Software. In *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion* (CSCW '16 Companion), 29–32. https://doi.org/10.1145/2818052.2874317

19. Sabine Graf and Prof. Kinshuk. 2006. An Approach for Detecting Learning Styles in Learning Management Systems. In *Proceedings of the Sixth IEEE International Conference on Advanced Learning Technologies* (ICALT '06), 161–163. Retrieved from http://dl.acm.org/citation.cfm?id=1156068.1156382

20. Tovi Grossman and George Fitzmaurice. 2015. An Investigation of Metrics for the In Situ Detection of Software Expertise. *Hum.-Comput. Interact.* 30, 1: 64–102. https://doi.org/10.1080/07370024.2014.881668

21. Mona Haraty, Joanna McGrenere, and Andrea Bunt. 2017. Online Customization Sharing Ecosystems: Components, Roles, and Motivations. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing* (CSCW '17), 2359–2371. https://doi.org/10.1145/2998181.2998289

22. Björn Hartmann, Daniel MacDougall, Joel Brandt, and Scott R. Klemmer. 2010. What Would Other Programmers Do: Suggesting Solutions to Error Messages. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '10), 1019–1028. https://doi.org/10.1145/1753326.1753478

23. Lichan Hong and Ed H. Chi. 2009. Annotate Once, Appear Anywhere: Collective Foraging for Snippets of Interest Using Paragraph Fingerprinting. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '09), 1791–1794. https://doi.org/10.1145/1518701.1518976

24. William Jones. 2008. *Keeping Found Things Found: The Study and Practice of Personal Information Management: The Study and Practice of Personal Information Management*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

25. Juho Kim, Phu Tran Nguyen, Sarah Weir, Philip J. Guo, Robert C. Miller, and Krzysztof Z. Gajos. 2014. Crowdsourcing Step-by-step Information Extraction to Enhance Existing How-to Videos. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems* (CHI '14), 4017–4026. https://doi.org/10.1145/2556288.2556986

26. 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), 999–1010. https://doi.org/10.1145/2531602.2531644

27. Aleksandra Klašnja-Milićević, Boban Vesin, Mirjana Ivanović, and Zoran Budimac. 2011. E-Learning personalization based on hybrid recommendation strategy and learning style identification. *Computers & Education* 56, 3: 885–899.

28. Neal Krawetz. Kind of Like That. *The Hacker Factor Blog*. Retrieved June 15, 2017 from http://www.hackerfactor.com/blog/?/archives/529-Kind-of-Like-That.html

29. Benjamin Lafreniere, Tovi Grossman, and George Fitzmaurice. 2013. Community Enhanced Tutorials: Improving Tutorials with Multiple Demonstrations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '13), 1779–1788. https://doi.org/10.1145/2470654.2466235

30. Karim R Lakhani and Eric von Hippel. 2003. How open source software works: "free" user-to-user assistance. *Research Policy* 32, 6: 923–943. https://doi.org/10.1016/S0048-7333(02)00095-1

31. Wei Li, Tovi Grossman, and George Fitzmaurice. 2014. CADament: A Gamified Multiplayer Software Tutorial System. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems* (CHI '14), 3369–3378. https://doi.org/10.1145/2556288.2556954

32. Lena Mamykina, Bella Manoim, Manas Mittal, George Hripcsak, and Björn Hartmann. 2011. Design Lessons from the Fastest Q&A Site in the West. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '11), 2857–2866. https://doi.org/10.1145/1978942.1979366

33. Justin Matejka, Tovi Grossman, and George Fitzmaurice. 2011. IP-QAT: In-product Questions, Answers, & Tips. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (UIST '11), 175–184. https://doi.org/10.1145/2047196.2047218

34. Justin Matejka, Wei Li, Tovi Grossman, and George Fitzmaurice. 2009. CommunityCommands: Command Recommendations for Software Applications. In *Proceedings of the 22Nd Annual ACM Symposium on User Interface Software and Technology* (UIST '09), 193–202. https://doi.org/10.1145/1622176.1622214

35. Dan Morris, Meredith Ringel Morris, and Gina Venolia. 2008. SearchBar: A Search-centric Web History for Task Resumption and Information Re-finding. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '08), 1207–1216. https://doi.org/10.1145/1357054.1357242

36. Meredith Ringel Morris, Jaime Teevan, and Katrina Panovich. 2010. What Do People Ask Their Social Networks, and Why?: A Survey Study of Status Message Q&a Behavior. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '10), 1739–1748. https://doi.org/10.1145/1753326.1753587

37. Dana Movshovitz-Attias, Yair Movshovitz-Attias, Peter Steenkiste, and Christos Faloutsos. 2013. Analysis of the Reputation System and User Contributions on a Question Answering Website: StackOverflow. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining* (ASONAM '13), 886–893. https://doi.org/10.1145/2492517.2500242

38. Emerson Murphy-Hill, Rahul Jiresal, and Gail C. Murphy. 2012. Improving Software Developers' Fluency by Recommending Development Environment Commands. In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of*

Software Engineering (FSE ’12), 42:1–42:11. https://doi.org/10.1145/2393596.2393645

39. Michael Nebeling, Maximilian Speicher, and Moira C. Norrie. 2013. CrowdAdapt: Enabling Crowdsourced Web Page Adaptation for Individual Viewing Conditions and Preferences. In *Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (EICS ’13), 23–32. https://doi.org/10.1145/2494603.2480304
40. Solomon Negash, Terry Ryan, and Magid Igbaria. 2003. Quality and Effectiveness in Web-based Customer Support Systems. *Inf. Manage.* 40, 8: 757–768. https://doi.org/10.1016/S0378-7206(02)00101-5
41. Dan R. Olsen Jr., Trent Taufer, and Jerry Alan Fails. 2004. ScreenCrayons: Annotating Anything. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology* (UIST ’04), 165–174. https://doi.org/10.1145/1029632.1029663
42. Oracle Corporation. MySQL :: MySQL Internals Manual :: 10.7 Full-Text Search. Retrieved July 22, 2017 from https://dev.mysql.com/doc/internals/en/full-text-search.html
43. Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work* (CSCW ’94), 175–186. https://doi.org/10.1145/192844.192905
44. Upendra Shardanand and Pattie Maes. 1995. Social Information Filtering: Algorithms for Automating “Word of Mouth.” In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI ’95), 210–217. https://doi.org/10.1145/223904.223931
45. V. Singh, M. B. Twidale, and D. M. Nichols. 2009. Users of Open Source Software - How Do They Get Help? In *2009 42nd Hawaii International Conference on System Sciences*, 1–10. https://doi.org/10.1109/HICSS.2009.489

46. V. Singh, M. B. Twidale, and D. Rathi. 2006. Open Source Technical Support: A Look at Peer Help-Giving. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06)*, 118c–118c. https://doi.org/10.1109/HICSS.2006.370

47. Vandana Singh and Michael B. Twidale. 2008. The Confusion of Crowds: Non-dyadic Help Interactions. In *Proceedings of the 2008 ACM Conference on Computer Supported Cooperative Work* (CSCW '08), 699–702. https://doi.org/10.1145/1460563.1460670

48. Anselm Strauss and Juliet Corbin. 1990. *Basics of qualitative research*. Newbury Park, CA: Sage.

49. Max G. Van Kleek, Michael Bernstein, Katrina Panovich, Gregory G. Vargas, David R. Karger, and MC Schraefel. 2009. Note to Self: Examining Personal Information Keeping in a Lightweight Note-taking Tool. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '09), 1477–1480. https://doi.org/10.1145/1518701.1518924

50. Laton Vermette, Parmit Chilana, Michael Terry, Adam Fourney, Ben Lafreniere, and Travis Kerr. 2015. CheatSheet: A Contextual Interactive Memory Aid for Web Applications. In *Proceedings of the 41st Graphics Interface Conference* (GI '15), 241–248. Retrieved from http://dl.acm.org/citation.cfm?id=2788890.2788933

51. Alan M. Wilson, Valarie A. Zeithaml, and Mary Jo Bitner. 2012. *Services Marketing: Integrating Customer Focus Across the Firm*. McGraw-Hill Higher Education.

52. Heather Wiltse and Jeffrey Nichols. 2009. PlayByPlay: Collaborative Web Browsing for Desktop and Mobile Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '09), 1781–1790. https://doi.org/10.1145/1518701.1518975

53. V. A. Zeithaml, M. J. Bitner, and D. D. Gremler. 2009. *Services Marketing: Integrating Customer Focus Across the Firm*. McGraw-Hill Irwin. Retrieved from https://books.google.ca/books?id=ODMlQAAACAAJ

54. 2017. *tesseract: Tesseract Open Source OCR Engine (main repository)*. tesseract-ocr. Retrieved from https://github.com/tesseract-ocr/tesseract

55. Stack Overflow - Where Developers Learn, Share, & Build Careers. Retrieved July 25, 2017 from https://stackoverflow.com/

56. Quora - A place to share knowledge and better understand the world. Retrieved July 25, 2017 from https://www.quora.com/

57. Super User. Retrieved July 25, 2017 from https://superuser.com/

58. Microsoft OneNote. Retrieved July 25, 2017 from https://www.onenote.com/

59. Video Editor - YouTube. Retrieved July 25, 2017 from https://www.youtube.com/editor

60. Upload audio or image files - YouTube Help. Retrieved July 26, 2017 from https://support.google.com/youtube/answer/1696878?hl=en

61. sklearn.feature_extraction.text.TfidfVectorizer — scikit-learn 0.18.2 documentation. Retrieved August 1, 2017 from http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html#sklearn.feature_extraction.text.TfidfVectorizer.fit_transform

62. sklearn.tree.DecisionTreeClassifier — scikit-learn 0.18.2 documentation. Retrieved August 1, 2017 from http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

63. Cleanslate. Retrieved July 25, 2017 from http://cleanslatecss.com/

64. CSS Cascading and Inheritance Level 4. Retrieved July 25, 2017 from https://www.w3.org/TR/css-cascade-4/#valdef-all-revert