Bachelor Thesis

August 23, 2023

ScreenCurator

Curation of digital knowledge with screenshots

Christoph Bachmann

of Rapperswil SG, Switzerland (20-701-090)

supervised by

Prof. Dr. Thomas Fritz

Alexander Lill, Roy Rutishauser

University of Zurich UZH

HASEL

Bachelor Thesis

ScreenCurator

Curation of digital knowledge with screenshots

Christoph Bachmann

University of Zurich^UZH

HASEL

Bachelor Thesis

Author:	Christoph Bachmann, christoph.bachmann3@uzh.ch
URL:	https://github.com/HASEL-UZH/screen-curator
Project period:	06.03.2023 - 06.09.2023

Human Aspects of Software Engineering Lab
Department of Informatics, University of Zurich

Acknowledgements

I would like to thank Roy Rutishauser, Alexander Lill, and Dr. Thomas Fritz for the chance to write my thesis with the HASEL research group. I want to further thank them for the great support and precious feedback they provided in the process. I enjoyed the weekly meetings where we discussed what features would represent a useful addition to the application. I also want to thank all the study participants for taking the time to test the application and sharing valuable insights. A great thank you goes out to my family and friends, who always encouraged me during the writing. In the end, I want to express my gratitude to Shannon for her mental support and understanding.

Abstract

In today's world, knowledge workers are often overwhelmed by the vast amounts of information they encounter while carrying out their tasks. As a result, it is vital to develop effective strategies for efficiently reusing previously foraged information to minimize foraging effort. One of these strategies is information curation, which is the concept of keeping, managing, and exploiting foraged information. Existing prototypes that addressed this topic have mostly specific use cases, like web resource curation or task history curation. Only a few of them allow the capturing of cross-application settings. None of these prototypes are optimized to support users in information foraging tasks. They lack extensive retrieval functionality, semantic content analysis, and structuring options for curated assets. To fill this void, we designed and developed the ScreenCurator. Our application allows users to capture cross-application screen settings and store them with extensive metadata. This combination shall enable comprehensive retrievability and reusability of curated knowledge. To provide users with a simple and pleasant experience, the ScreenCurator implements a certain degree of automation combined with an intuitive interface. Our application was evaluated in a user study where seven participants used the ScreenCurator for 10-15 working days besides their daily tasks. The gathered feedback implied that our approach improved the experience of taking and retrieving screenshots. Furthermore, two high-level use cases could be identified: long-term backups and short-term to-dos. Nevertheless, we found that the ScreenCurator needs to increase the implemented degree of automation and add further structuring options. Additionally, it would be of great value if the ScreenCurator would enable collaborative curation and knowledge sharing. Besides extending the feature set, care should be taken to maintain the simplicity and intuitiveness of the application.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

We reached an era in which knowledge workers need to develop strategies to deal with personal information overload to be efficient [NK18]. They are confronted with a huge amount of information that is visually displayed on their screens. To be efficient, this knowledge should only be foraged once and then be easily retrievable in case it is needed again. An approach to minimize information retrieval costs is information curation, which ensures the long-term access, retrieval, and reuse of data [Poo17]. One problem of current personal digital curation solutions is that there is no single method for all types of resources [HL22]. So, for example, if a piece of knowledge results from the combination of a Microsoft Teams message and a code snippet in Visual Studio Code, you can currently only curate and annotate both on their own while losing information on their interconnection.

One option to capture such a setting would be to take a screenshot. Screenshots as a medium of information curation have significant advantages, such as simplicity and good recognizability. It requires low effort to take screenshots, while the advantages of visual processing apply. Humans can memorize not only the gist but also details of an extensive amount of images, which could later be useful to retrieve curated images and restore their context faster [BKAO08]. Often, the context of an activity or an event can be remembered more easily than the exact details and is, therefore, very useful to search for curated information [LF94]. However, the disadvantage of using simple screenshots as a storage medium is that much of the meta information, like timestamps, context, and textual information, is lost. Additionally, due to the very limited traceability of a simple image, it is not possible to search for it based on its content or context. Therefore, it is time-intensive and laborious to retrieve a screenshot, especially in a larger, non-hierarchical library. To address the shortcomings of simple screenshots, we devised an approach that combines its visual aspects with textual metadata of the semantic content and context.

There are a few prototypes that have already approached information curation as well by screen capturing. One of them is Scrapbook [HL22], a screenshot application that collects open windows and allows the user to set a title and a description for each screenshot. Fuse [KCH^{+}22] is a browser-specific sidebar that stores screenshots of collected websites together with an URL in a hierarchically structured library. ScreenTrack [HL20] is an application that captures a user's screen continuously, records the front-most application, and creates a visual history from it. However, each of these prototypes offers few automated metadata collection features and limited retrieval options. Moreover, most existing prototypes target a special niche, like web resource curation.

1.2 Project

To the extent of this thesis, an explorative prototype of a digital screen curator is designed and implemented. This prototype was constructed to tackle the challenges of information curation by screenshots previously described. This prototype is referred to as ScreenCurator in the course of this thesis. This application enables users to easily capture screenshots of relevant digital information during their search activities. Screenshots are then saved together with metadata in a structured library. A dynamic tagging system provides the optimal structure for the user to sort screenshots into projects and their context. With this tagging structure, users can easily group screenshots based on their context, uncover additional connections between selected items, and search for specific elements by scrolling through the collection. For each screenshot, metadata, including a timestamp and visible applications, are collected to reconstruct the surrounding context and ease the users' search for an element. To further increase the traceability and usability of the collected screenshots, visible text elements are extracted. All of the associated data helps the user to find screenshot elements faster. Furthermore, the prototype enables the use of keyword searches and attribute filtering on the screenshot collection. The search function helps the user to efficiently reuse information and reduce the time and effort needed for information retrieval. The ScreenCurator prototype was developed in an iterative and agile project context to optimally support the explorative process. This includes user stories as main working products and weekly meetings to determine further features and working steps.

In the end, the implemented ScreenCurator app was evaluated in the form of a user field study. A small group of users spent two to three weeks working with the ScreenCurator and were then able to give feedback regarding the functionality and usability of the implemented prototype. Based on this feedback, the design and the features of the ScreenCurator app were assessed. Additionally, a small Pre- and Post-Questionnaire was handed out to participants to detect the changes in the experience of working with screenshots while using the ScreenCurator.

This thesis' goal is to contribute to the field of personal digital information curation. A focus is set on cross-application resources that have a visual representation. The ScreenCurator shall offer a proof of concept and a platform to analyze this form of curation. It is intended to discover if such an application changes the experience of using screenshots as a curation medium (RQ1). Furthermore, the specific use cases (RQ2) and the needed set of features (RQ3) shall be clarified by this thesis. Consequently, the research questions that shall be answered are:

- RQ1: How does the ScreenCurator change the experience of using screenshots as a medium of information preservation?
- RQ2: In what situation and for which type of data is digital knowledge curation through screenshots useful and efficient?
- RQ3: Which features should the ScreenCurator app provide to optimally support the storage and retrieval of digital knowledge?

Chapter 2

Related Work

In this part, an analysis of existing tools and theoretical work is conducted, which is interconnected with the topic of this thesis. This knowledge will serve as a basis for the development of the ScreenCurator prototype. First, theoretical concepts are analyzed, including information foraging and -curation, as well as visual processing. General workflows and definitions of information foraging and -curation are reviewed. Moreover, the limitations and possibilities of visual processing are analyzed. Secondly, manual as well as assisted approaches to information curation are explored. Many of the assisted curation approaches focus on a specific type of information or type of metadata. Many of them also have limitations when it comes to the retrieval of curated information.

2.1 Information Foraging & Curation

To enable the reuse of information, it is crucial to have a good understanding of how information is sought and stored. In today's working environment, using various types of information is crucial for knowledge workers. It's important to handle this information with care while also considering the costs and benefits that arise in the retrieval process. Pirolli [Pir07] describes the theory that humans in the digital age similarly seek information and knowledge as they used to forge for food in the earlier days. It is stated that an optimal foraging process is always a tradeoff between seeking and selection cost against the benefit of a found element. So, to ensure the efficiency of modern knowledge workers, the cost of information foraging should be as low as possible. Whittaker [Whi11] expands the concept of information foraging with a focus on personal information repositories instead of public resources. He argues that this familiar information has to be kept, managed, and exploited as part of its life cycle. These activities bring their costs and benefits, which have to be taken into account. He calls this life cycle process "information curation". To be effective, the ScreenCurator must pay close attention to this life cycle.

In the digital age, the curation of knowledge has become a widely applicable field of research. There are multiple definitions and variations of how the keeping, managing, and exploitation shall happen. According to North et al. [NK18], curation means to maintain and add value to a collection of knowledge. Pennock [Pen07, p.1] in turn comes up with the following definition: "Digital curation ... is about maintaining and adding value to a trusted body of digital information for both current and future use: in other words, it is the active management and appraisal of digital information over its entire life cycle". So, digital curation helps users to handle the vast amount of information that they are confronted with. This exactly marks the ScreenCurator's right to exist. Heidorn [BH11] does not state an explicit definition, like the other two, but outlines the importance of proper curation. He writes that when data collection in a research project starts, it is vital to ensure a proper format and to collect metadata at the same time. Moreover, he writes that the reuse of research data of a project contributes as much as the reuse of a discovered concept. This supports the idea of the ScreenCurator to offer a place where discovered concepts, information, and data can be kept in a uniform format together with metadata.

Information curation in knowledge-based projects is not only vital for individuals but also for the team as a whole and, therefore, the entire project. Kamara et al. [KACB03] describe the use of a central project knowledge file together with a workflow system and a knowledge manager to ensure the efficient reuse of information inside a digital project. This central collection, maintenance, and reuse of project data can be seen as a form of information curation. The palette of different data that is created and reused by knowledge workers in a project context was analyzed by Poole [Poo17]. They found that web data, metadata, presentations, algorithms, maps images, code, and many other types of data were used, and even the great shift towards less traditional data has not occurred yet. Different formats of data increase the difficulty of curating them together. Most of the time, different file types, like code files and Word documents, are accessed with specific applications. This prevents the user from taking advantage of the application's internal bookmarking options if the user does not want to lose information about the relation between different data types. This kind of cross-application information curation is supported and even simplified by the ScreenCurator.

Bookmarking can be used as a form of digital information curation. So, for example, bookmarking a website means keeping a link to previously used information. This decreases the cost of further retrieval as well. Abrams et al. [ABC98] describe bookmarks on page 41 as "...historical pointers to useful information that may otherwise be forgotten". It is written that personal information systems, like bookmarking systems, serve as a strategy to reduce information overload and present only a focused, quickly accessible subset of highly relevant information to a user. They further mention that the organization of bookmarks is a tradeoff between managing cost and retrieval cost. Duc-Hieu [Tra23] presents the idea of augmenting screenshots with semantic- and contextual metadata to use them as visual bookmarks. It is argued that different techniques like image processing and optical character recognition can be used to generate such metadata, as well as a semantic index. A dense vector representation of the metadata is proposed to make it suitable for further processing steps. Apart from the theoretical idea, the paper does not present any kind of implementation. Nevertheless, the proposed concept is already very close to the idea of the ScreenCurator and is, therefore, highly relevant. Furthermore, it confirms the need and usefulness of such an application.

2.2 Visual Processing

To further elaborate on the usefulness of the ScreenCurator, it is important to take a look at human visual capacities. Unlike text, images can be processed in parallel by the human brain [WGK10]. This means that visual information can, in some cases, be processed significantly faster than textual information. Especially when viewing an image preconscious, like it happens when a user is scrolling through a gallery, perception works automatically and is preattentive [WGK10]. This makes visual information a valid medium for effective information curation. A glance at what the human visual long-term memory is capable of, in terms of storage, was shown by Brady et al. [BKAO08]. They found that the visual memory was not exhausted by a set of 2,500 detailed representations, while observers successfully remembered the images' details after a single viewing. The human brain's ability to recall intricate details within a depicted scenario is a valuable prerequisite for memory retention. The results of Ding et al. [DMS99] show that the combination of visual and textual information is especially efficient. They found that verbal information can help users extract more information from an image, improve comprehension, and decrease the required processing time. Their findings underline the effectiveness of the combination of textual metadata and screenshots. Regarding different types of visual information, it was also shown that static screenshots generate more accurate contextual recall than animations do [RTBH15]. This approves our decision to use screenshots instead of short screen recordings to curate a screen setting.

2.3 Manual Information Curation

There are several ways how knowledge workers can curate found information and reuse it in a non-automated fashion. Nevertheless, the lack of automation causes extra work due to the need to keep curated resources organized and traceable. This, in turn, increases the cost of keeping resources and makes it less likely that users curate a diverse amount of information.

2.3.1 Note-Taking

Note-taking tools like Microsoft OneNote or Evernote can be used for content curation and, therefore, also for information curation [Dal14]. These tools help to organize and save notes in a way that allows users to retrieve and reuse them again. Information content can be manually added in the form of text, images, or sketches. This information is then saved in a notebook structure and can be retrieved via global search. Therefore, they offer a free-form curation space that allows spatial organization of content [KLL$^+$17]. Kerne et al. [KLL$^+$17] argue that free-form curation can exploit the potential presented by new media channels and interaction modalities. As a result, free-form curation also enables the curation of information in visual form since screenshots can just be added and annotated. Chuter [Chu] writes that OneNote provides a personally curated digital space that can be used to recall and rediscover ideas, as well as materials. This shall enable more efficient work. One issue with this type of manual curation is that various types of metadata must be added separately. If not organized well, this can lead to having vital information all over the place, which results in more effort while retrieving the curated information. Furthermore, it is often laborious to manually add comprehensive contextual information.

The ScreenCurator app, in turn, stores all metadata together with the corresponding image, which forms small information units. This smaller information unit shall help to retrieve and process curated resources more quickly. Moreover, these small information units are not stored in notebooks but are organized with a tagging system, which allows to indicate cross-element dependencies. The visual focus of the ScreenCurator, together with a small but effective feature set and a clear workflow, shall help the user to not get lost in the curation process. Therefore, ScreenCurator offers a more focused and straightforward curation solution than note-taking tools do.

2.3.2 Software Development Artifact Curation

There exist multiple tools that focus on the curation of programmatic assets. One of these tools is GitHub Gist [Cre]. These GitHub repositories can be used to share or store small components to make them reusable [WPCWG15]. Therefore, they can be seen as an information curation tool in the field of software engineering. According to Wang et al. [WPCWG15], not only files in programming language formats are stored in Gists but also many files in Markdown and JSON. They infer that a lot of text data, such as documentation and logs, are stored in Gists. Another tool in this field is Pastebin [Pasb]. It also allows the storage and sharing of programmatic assets like code snippets. Compared to Gists, it also encourages the pasting of various types of texts, like notes or ideas. Furthermore, it allows the user to structure pastes in folders, in categories, and by adding tags [Pasa].

The ScreenCurator is designed to assist developers with their research tasks, although it is not primarily intended for curating programmatic assets. However, one of the core ideas is to enable the curation of a combination of code snippets and additional information resources in different formats. In our case, the resulting knowledge is the interesting asset.

2.4 Assisted Information Curation

Assisted information curation applications implement a certain degree of automation in the information curation life cycle. There exist several prototypes that have similar applications in the field of personal information curation to the ones of the ScreenCurator. Following this, the most important approaches are analyzed and compared to the ScreenCurator prototype. These approaches helped to outline the initial feature set and served as a reference to how curation happens in practice.

2.4.1 Speech Data Curation

A special kind of metadata is speech data. Speech data can be used for all kinds of audio notes, descriptions, or reasoning. Therefore, it has a valid application in the field of personal curation in an alternative multimedia format.

Co-Notate [ROH19] is, for example, a prototype that allows video and audio recording of a design process. Recordings can be augmented by adding short notes or user-predefined tags on button-click. Rasmussen et al. [ROH19] writes that after a session, various views of the archived data are presented to the user.

Another application that makes even more use of metadata in spoken form is *Think-Aloud Computing* [KAM$^+$21]. Here, not only the session but all of the user's spoken-out thoughts during a working process are recorded to curate contextual knowledge about a project. The *Think-Aloud Computing* prototype [KAM$^+$21] combines speech recording with screen recording, action recording, and optional textual comments. Speech recordings are automatically transcribed. Krosnick et al. [KAM$^+$21] also mention the possibilities of search and filter options that shall help to retrieve specific information.

Like the described prototypes [KAM$^+$21] [ROH19], the ScreenCurator is built to capture not only simple images of screen settings but also semantic contextual information. Different from the speech data curation prototypes, the ScreenCurator app focuses not on explicit spoken context information but rather on the implicit context of the visual content, like open applications. Furthermore, the user's actions are not in the spotlight of the concept of ScreenCurator, contrary to *Think-Aloud Computing*.

2.4.2 Digital Task History Curation

Another type of personal curation focuses on capturing the user's journey and history while navigating between tasks in a digital space. The reviewed prototypes do this through an automated form of continuous state capturing. This form of personal curation can help a user regain task state knowledge or reuse previously applied procedures.

A project that focuses on general capturing and retrieval of screen information of users' digital journeys is *ScreenTrack* [HL20]. This tool regularly takes screenshots to create a visual history in the form of a time-lapse. It captures the frontmost application and the corresponding file path. *ScreenTrack* is available as a browser plug-in. The results showed that visual bookmarks helped participants to reconstruct mental context and reopen relevant resources [HL20].

Satterfield et al. [SFM20] introduced an approach that captures the user's digital workspace in regular intervals as well. They state to extract text elements with the Tesseract OCR and then generate a list of all words. These 'bags of words' are then mapped to tasks, which allows them to create a collection of user tasks. This would allow users to review all tasks they have worked on during the day. According to them, this approach can also be used to find resources that were used to the extent of a task. In the end, they could capture tasks with an accuracy of 70.6% [SFM20].

Like the presented digital task history curation approaches [SFM20] [HL20], the ScreenCurator app enables the user to find previously accessed and used resources again. Like both approaches, the ScreenCurator only works for content that is visual. Similar to the approach of Satterfield et al. [SFM20], the ScreenCurator uses OCR to extract text elements and use them as semantic metadata. Different from the task history curation prototypes, the ScreenCurator uses less automation and, therefore, increases precision. This lowers the user's search effort as well as memory requirements. Hu et al. [HL20] found that each application has its own kind of metadata that is important to the user, which is also relevant for the ScreenCurator. Their mentioned concerns about the user's data privacy should be taken into account, as well. The proposed solution to store user data only locally was also implemented in the ScreenCurator.

2.4.3 Web Resource Curation

Web Resource Curation prototypes focus on capturing and reusing web research results. The two analyzed prototypes come as browser extensions and have more specific use cases. Nevertheless, especially in the area of information foraging, internet browsers are omnipresent nowadays and, therefore, of great interest for an effective knowledge curation solution.

Crystalline [LKM22] is a prototype that automatically collects and organizes information while the user browses the internet. The application injects a sidebar into every web page and provides the user with an automatically collected list of encountered options and criteria that a user has looked into. The collection happens through the detection of user signals, like mouse movement, and analysis of web-page's HTML. A comparison table is built to visualize options and criteria and support the user's decision-making.

Another sidebar-based browser extension is *Fuse* [KCH$^+$22]. It aims to lessen the divide between searching for, collecting, and organizing information. In comparison to *Crystalline*, it implements less automation. Moreover, it allows users to create projects in which different web resources can be imported. The system saves data about collected items, such as URLs, screenshots, and favicons. *Fuse* [KCH$^+$22] supports a hierarchical structure of nesting of web resources while displaying a condensed representation. In a 22-month-long user study, *Fuse* was tested in personal, as well as professional tasks.

Like the web resource curation prototypes [KCH$^+$22] [LKM22], the ScreenCurator app is thought to support people in situ in information foraging tasks. Moreover, it shall help the user to minimize the cost of searching, collecting, and organizing information together with metadata. However, the ScreenCurator focuses on cross-application resources to enable greater freedom in the information collection process. Furthermore, the ScreenCurator app stores additional semantic metadata, such as textual screenshot content. Different from the ScreenCurator app, which implements semi-automatic information collection, *Crystalline* has fully automatic information collection. Therefore, the ScreenCurator is more precise but also has a higher collection cost.

Certainly, there are many interesting prototypes in the field of web bookmarking. Nonetheless, concrete prototypes are considered less important in this context due to the lack of elements that are stored in visual form.

2.4.4 Screenshot-based Bookmarking

The existing prototype, which comes closest to the implemented ScreenCurator prototype, is *Scrapbook* [HL22], which allows taking screenshots, gathering corresponding metadata, and storing everything in a global collection. According to Hu et al. 2022 [HL22], the goal of Scrapbook is "...to assist knowledge workers with the process of reconstructing their working context by allowing them to curate cross-application resources".

Scrapbook allows the user to store screenshots together with a title, a description, and a timestamp. Moreover, it collects all application windows that are visible on a screenshot. The application offers a collection view, where all taken screenshots are visible. To identify visible windows, *Scrapbook* implements a dynamic bit-masking algorithm to identify the subset of visible windows in the set of open applications. This algorithm uses two-dimensional arrays containing 1s and 0s to represent the area of open application windows. *Scrapbook* is implemented as a window-less application for Mac OS. It uses an operating-system-level accessibility API to collect metadata such as timestamps and open applications.

Like *Scrapbook* [HL22], the ScreenCurator app implements functionality to take screenshots and collect metadata to capture the image's context. An improvement that was proposed in *Scrapbook* [HL22] is to categorize short-term - and long-term resources. This provides better support for reminder-like screenshots and helps to keep the collection at a reasonable size. This problem is solved by the ScreenCurator, which offers users a dynamic tagging structure. Tags allow the temporal usage categorization of curated elements. Additionally, the tagging structure enables the user to indicate relations and dependencies between screenshots, which is not possible with Scrapbook. This tagging option is one of the foundations for the filtering of screenshots, which is exclusively implemented by ScreenCurator. Different from Scrapbook, the ScreenCurator app focuses on information foraging tasks and implements corresponding features, like a free-hand-drawing option and text extraction. Whereas Scrapbook uses almost only spelled-out text in its interface, the ScreenCurator mainly makes use of graphical items, such as icons, in its interface. This will help to make use of the advantages of visual processing, not only for information retrieval but for the interface itself. Moreover, the ScreenCurator prototype will be implemented not only for Mac OS but also for Windows.

2.4.5 Summary

The introduced web research curation prototypes [KCH$^+$22] [LKM22] focus completely on a web setting and come as a browser extension. Therefore, they are not able to capture cross-application information. Two of the other prototypes have the core idea of documenting the user's digital task history [SFM20] [HL20]. Since these approaches use continuous screen capturing, their output is very broad and too imprecise for effective curation in the field of information foraging. Additionally, the large output offers limited scalability and searchability. The speech data curation prototypes [KAM$^+$21] [ROH19] have a special focus on spoken-out contextual thoughts, which is not the case for the ScreenCurator. Both *Think-Aloud Computing* [KAM$^+$21] and *ScreenTrack* [HL20] additionally contain a videographic data format, which is, due to poor scalability, not in the focus of the ScreenCurator. Furthermore, according to Rule et al. [RTBH15], videographic animations cause less accurate context recall than static screenshot thumbnails. *Scrapbook* [HL22], offers a screenshot-based, cross-application curation solution. Nevertheless, it provides limited support for retrieval, except for a search bar. Moreover, there is no possibility to structure stored elements, which could further help during retrieval situations. Most of the screenshot's semantic content, such as contained text elements, is ignored by *Scrapbook*, which leads to limited traceability. Additionally, the options to augment captured information leave room for enhancements since there are no options to mark important sections or categorize the information.

Chapter 3

Approach

To the extent of this thesis, ScreenCurator, a prototype of a cross-application information curation app, was implemented. In the following chapter, we further describe the prototype, its high-level concepts, and the corresponding design decisions. Moreover, it is explained how metadata is collected, how the tagging system works, and how the retrieval functionality is implemented. In the end, design decisions and features are presented that shall enhance the application's usability.

3.1 Concepts

The fundamental requirements for the ScreenCurator were that it offers an intuitive platform to store and retrieve knowledge in a visual format. It shall provide a semi-guided workflow to reduce complexity and implement Whittaker's [Whi11] curation life cycle. Therefore, the application shall support the keeping, managing, and exploitation of information curated in the form of screenshots. Furthermore, the use-case focus of the ScreenCurator was set on general information foraging tasks. Additionally, the ScreenCurator shall follow usability standards and be dynamic as well as scalable.

3.1.1 Emphasis Visual Representation

The ScreenCurator focuses on the curation of resources with a static visual format. This format optimally implements the previously mentioned advantages of visual processing. Therefore, screenshots offer a very convenient and low-effort way of curating various resources with a visual representation. Consequently, the primary curation data format is visual, while the metadata's textual format is secondary. The user only has to mark the area of interest by spanning a rectangle. This feature allows the user to accurately select what they want to curate and reduce the amount of effort needed to retrieve it.

3.1.2 Cross-Application Functionality

It is an important attribute that the application supports the curation from numerous digital sources. Therefore, the ScreenCurator offers great flexibility due to its focus on visual representations. However, it also offers the possibility to add application-specific information that is not visible to captured elements. This information can be added as metadata in the form of a custom description. The concept of cross-application information capturing is further supported through the differentiation of various sources. This happens through the automated recognition of captured applications, which are stored as metadata for each curated screenshot. In turn, all of the mentioned metadata provides a valuable basis for the further retrieval of screenshots.

3.1.3 Semi-automated

The prototype offers the users a mix of automation and manual customization of captured information. As soon as an area of interest is selected, the ScreenCurator provides suggestions for multiple types of metadata. Nonetheless, it allows the user to customize all of them. By utilizing this method, users can save time and effort in curating resources while also having the opportunity to enhance the accuracy of metadata. As a result, the user can outweigh the cost of custom modification of metadata and the benefit of better retrievability of the screenshot for each element individually.

3.1.4 Scalability & Management Support

The ScreenCurator is built to be able to handle even large amounts of screenshots. Not only are the technical foundations in place to handle large quantities of data but also the feature set allows the handling of a growing collection. With extended retrieval capabilities such as text search, users can instantly retrieve items from both small and large collections. The collection's structuring options further support this concept. To help users to uphold the relevance of screenshots in the library, the application offers functionality to identify short-lived elements like To-Do items. You can easily find items that haven't been organized within the application structure. Additionally, the application provides several options to delete or tidy up these screenshots.

3.1.5 Screenshot Retrievability

All screenshots in this application shall be precisely trackable and retrievable. To enable this, multiple types of metadata are stored together with screenshots. The metadata builds the basis of extended search- and filter functionalities. These retrieval functionalities, in turn, enable the user to track down individual screenshots effortlessly. This ensures the efficiency of retrieval and the entire curation life cycle. The combination of screenshots with metadata and strong search functionalities shall ensure low-cost retrieval of visual resources, which normally provide low retrievability.

3.1.6 Intuitive & Simple Usage

To make the ScreenCurator attractive to potential users, it has to implement good usability. The interface should follow the same principles as the app's underlying idea, utilizing the benefits of visual processing whenever possible. Therefore, the interface makes heavy use of icons and colors. Due to this, the ScreenCurator provides an intuitive interface that implements visual variables such as colors and shapes. Furthermore, attention was paid to providing an enclosed and clear feature set. Lastly, shortcuts shall offer users a way to increase their efficiency when using the application.

3.2 Application Workflow

As mentioned before, the application's workflow is inspired by the curation life cycle of Whittaker [Whi11]. It is further separated into the keep workflow and the exploitation workflow. The curated elements are managed in both workflows. The keep workflow starts when a user decides to take a screenshot. The user will then be able to select the area of interest containing a piece of knowledge that should be kept. After confirming the area of interest, the captured knowledge can be refined and configured. Eventually, the user can decide if an element is stored or discarded. As soon as one or multiple items have been curated, the user can start the exploitation workflow. The users can then apply filters, keyword searches, and scrolling searches to regain curated resources. The visual nature of the screenshots shall help the user to quickly locate the desired piece of knowledge in a group of different results. Once the searched element is located, the user can visually extract and reuse the curated knowledge. The application also allows direct resource reuse by copying the image or extracted text. The user can then manage and maintain the resource by updating most of the curated element's metadata. The curated knowledge can further be updated or augmented by the freehand-drawing option. This management and maintenance part is vital to keep the resources relevant and reusable. Because after taking a screenshot, one is directed back to the same location in the application, these two workflows can happen in parallel. The only dependency of these two workflows is the metadata. It is assigned during capturing and later used in the searching process during exploitation. With three main actions each, these workflows are held deliberately lean to reduce the overhead of the application.

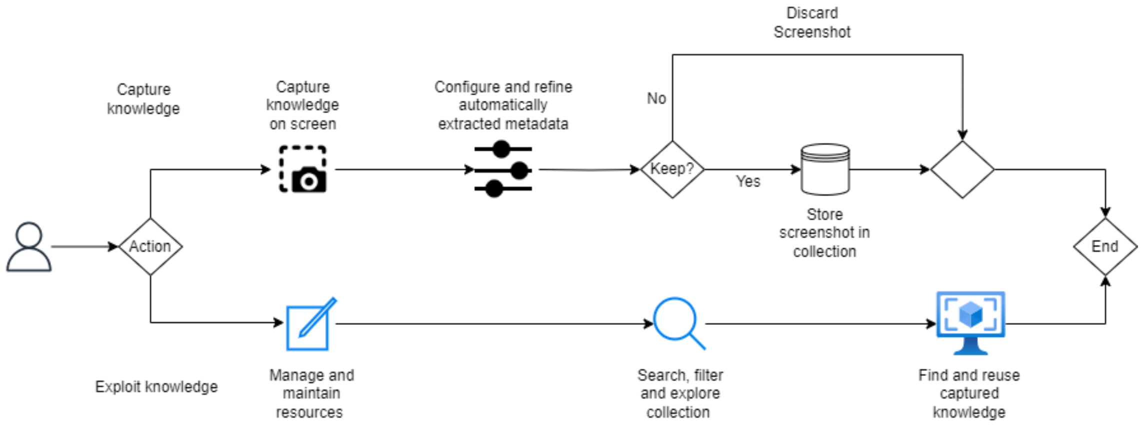

Discard Screenshot
Capture knowledge
Capture knowledge on screen
Configure and refine automatically extracted metadata
No
Keep?
Yes
Store screenshot in collection
Action
End
Exploit knowledge
Manage and maintain resources
Search, filter and explore collection
Find and reuse captured knowledge

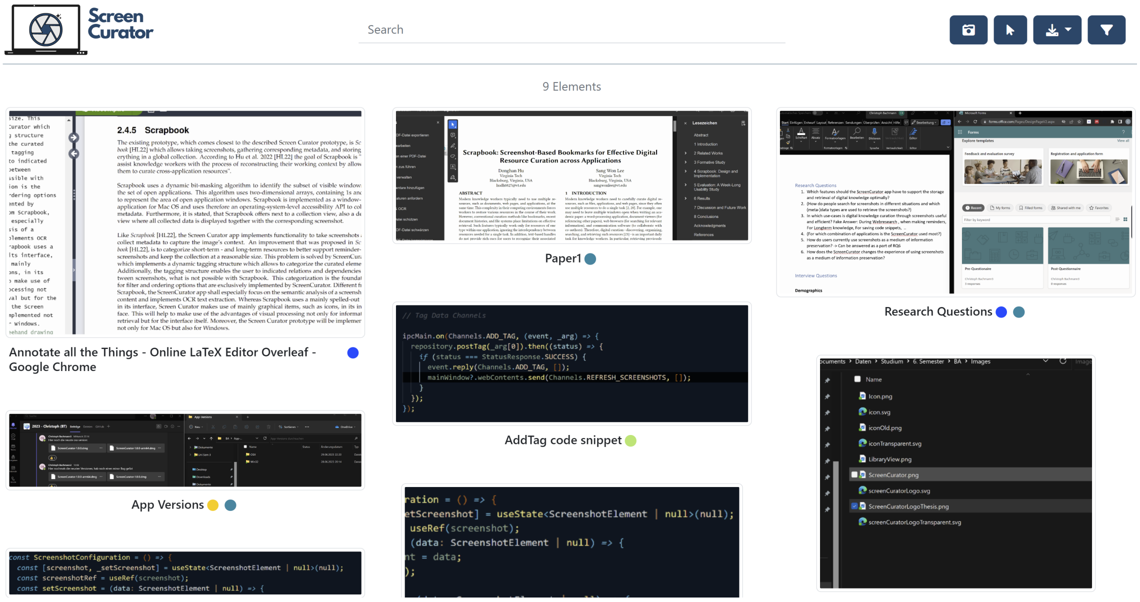
Figure 3.1: Workflow Diagram

3.3 Prototype

3.3.1 Application Overview

The ScreenCurator is implemented as a basic Electron [Bui] desktop application. It consists of a screenshot view, a library view, and a detail view to capture, retrieve, and maintain resources. Each of these views delivers a self-contained set of features.

The Library View is the entry point into the app. Here, the users see their collection of screenshots at first sight. This shall contribute to the application's simplicity and help to make it intuitive. Every screenshot is displayed together with its title and colored dots that indicate tags that are assigned to the screenshot. On hovering over the dots, the tag names are displayed in appearing tool-tips. By only showing the associated color of the tags, space can be saved and visual processing advantages can be exploited. Moreover, there exists a feature on this view that allows users to export all screenshots. This enables the usage of the collection outside the application. Another feature of this view allows the user to perform operations on multiple screenshots simultaneously. These features are meant to increase the usability of the ScreenCurator.



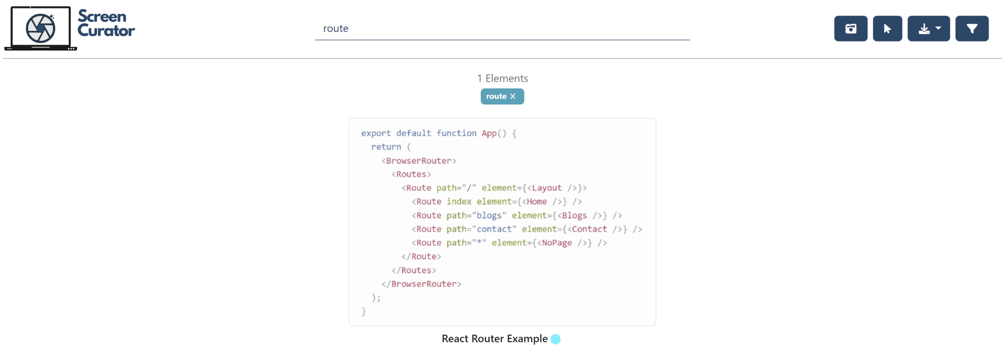
```
// Tag Data Channels

ipcMain.on(Channels.ADD_TAG, (event, _arg) => {
  repository.postTag(_arg[0]).then((status) => {
    if (status === StatusResponse.SUCCESS) {
      event.reply(Channels.ADD_TAG, []);
      mainWindow?.webContents.send(Channels.REFRESH_SCREENSHOTS, []);
    }
  });
});
```

```
const ScreenshotConfiguration = () => {
  const [screenshot, _setScreenshot] = useState<ScreenshotElement | null>(null);
  const screenshotRef = useRef(screenshot);
  const setScreenshot = (data: ScreenshotElement | null) => {
```

```
ration = () => {
etScreenshot] = useState<ScreenshotElement | null>(null);
 useRef(screenshot);
 (data: ScreenshotElement | null) => {
nt = data;
);
```

Figure 3.2: Library View

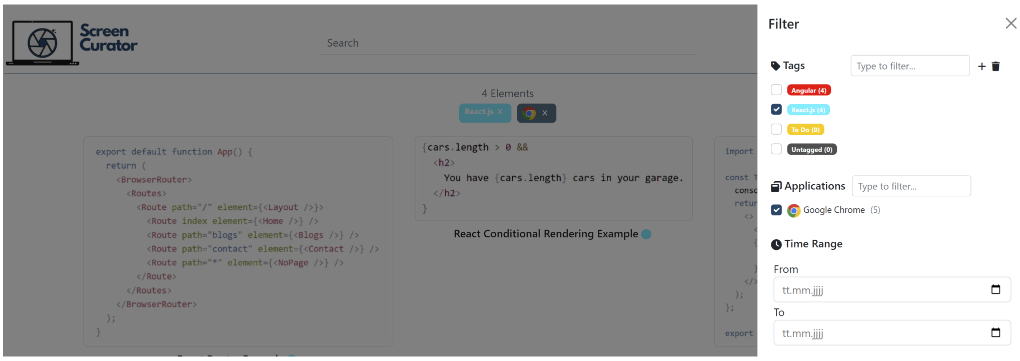
All search and filtering options can be found on the library view. They represent one of the core parts of the application due to their purpose of enabling the reuse of curated knowledge. The retrieval functionality provides a keyword search, as well as a tag-, a time range- and an application filter. This toolset is designed to be both easy to use and powerful. The keyword search targets all textual metadata fields. This includes the title, description, and extracted text of a screenshot. Additionally, the tag and the application name are targeted by the keyword search. On key-hit, every screenshot that contains a matching metadata field is shown. The filters allow the user to select tags and programs and view the corresponding screenshots in the library. Both the application and tag filters can be searched in a separate search box in the filter menu. The time range filter allows users to only select screenshots that were made in a certain time frame.



```
export default function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Layout />}>
          <Route index element={<Home />} />
          <Route path="blogs" element={<Blogs />} />
          <Route path="contact" element={<Contact />} />
          <Route path="*" element={<NoPage />} />
        </Route>
      </Routes>
    </BrowserRouter>
  );
}
```

React Router Example

Figure 3.3: Keyword Search Retrieval



```
export default function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Layout />}>
          <Route index element={<Home />} />
          <Route path="blogs" element={<Blogs />} />
          <Route path="contact" element={<Contact />} />
          <Route path="*" element={<NoPage />} />
        </Route>
      </Routes>
    </BrowserRouter>
  );
}
```

```
{cars.length > 0 &&
  <h2>
    You have {cars.length} cars in your garage.
  </h2>
}
```

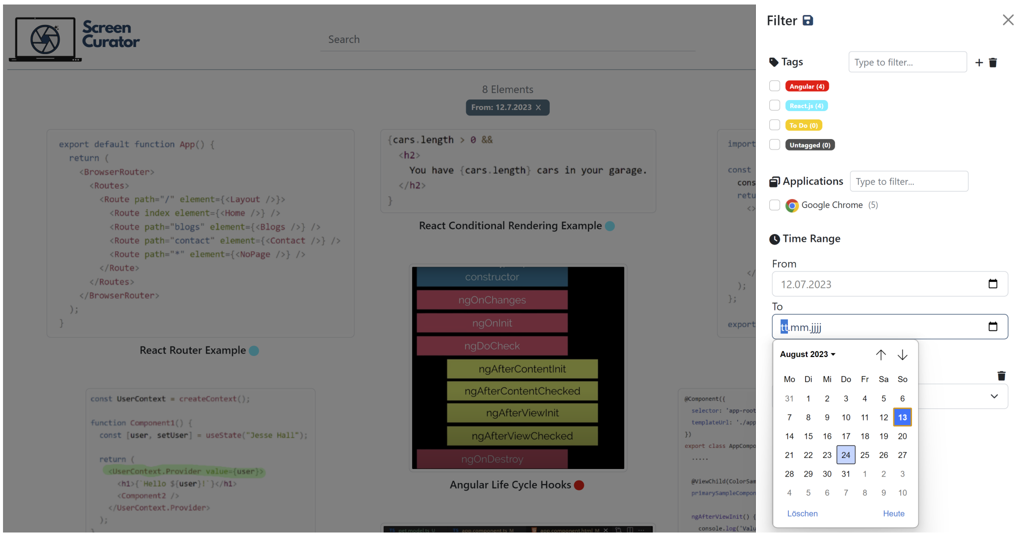
React Conditional Rendering Example

- ☐ Angular (4)
- ☑ React.js (4)
- ☐ To Do (0)
- ☐ Untagged (0)
- ☑ Google Chrome (5)

From: tt.mm.jjjj

To: tt.mm.jjjj

Figure 3.4: Tag- & Application Filter Retrieval



```
export default function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Layout />}>
          <Route index element={<Home />} />
          <Route path="blogs" element={<Blogs />} />
          <Route path="contact" element={<Contact />} />
          <Route path="*" element={<NoPage />} />
        </Route>
      </Routes>
    </BrowserRouter>
  );
}
```

React Router Example

```
{cars.length > 0 &&
  <h2>
    You have {cars.length} cars in your garage.
  </h2>
}
```

React Conditional Rendering Example

Angular Life Cycle Hooks

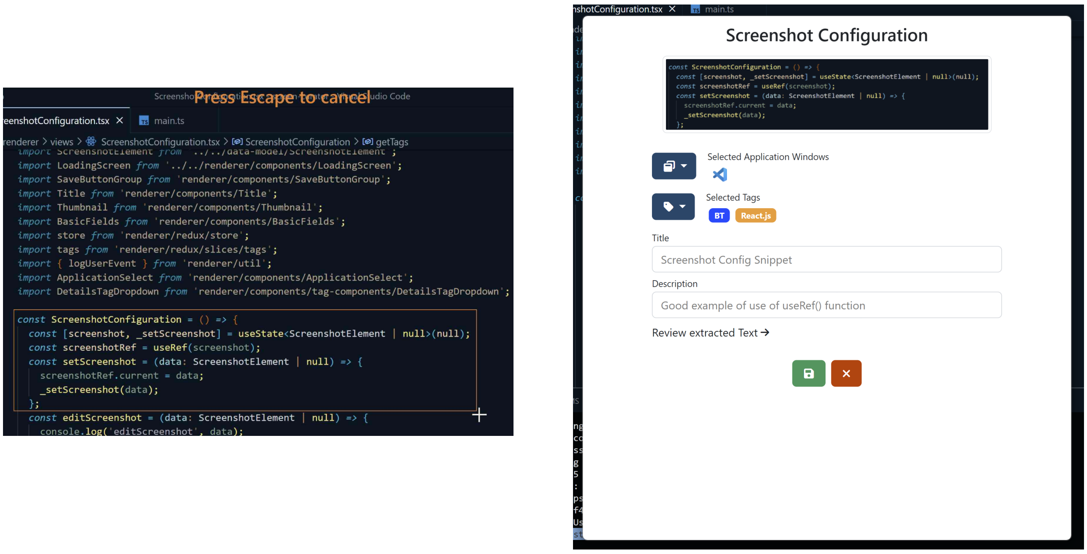
```
const UserContext = createContext();

function Component1() {
  const [user, setUser] = useState("Jesse Hall");

  return (
    <UserContext.Provider value={user}>
      <h1>{`Hello ${user}!`}</h1>
      <Component2 />
    </UserContext.Provider>
  );
```

From: 12.07.2023

To: tt.mm.jjjj

Figure 3.5: Time Range Filter Retrieval

When a user wants to curate a new element, the "Take Screenshot"-action can be performed. This happens by clicking a button on the library view or using a global keyboard shortcut. A global shortcut shall further decrease keeping costs and increase the usability of the ScreenCurator. The user is then redirected to the Screenshot view. Here, the user can select the area of interest by spanning a rectangle. On mouse release, the configuration view appears, where the user can augment the taken screenshot with a title, description, and tags. The implemented type of ’screen snipping’ is commonly used and known from multiple operating-system-integrated snipping tools. Additional metadata, such as text extracted from the image, open applications, and a timestamp, are added automatically to reduce the user’s workload. Nevertheless, the user can correct the assigned applications and the extracted text in the configuration view. This allows the users to always increase a screenshot’s retrievability.


Press Escape to cancel
eenshotConfiguration.tsx ×
main.ts
renderer > views > ScreenshotConfiguration.tsx > ScreenshotConfiguration > getTags

```
import ScreenshotElement from '../../data-model/ScreenshotElement';
import LoadingScreen from '../../renderer/components/LoadingScreen';
import SaveButtonGroup from 'renderer/components/SaveButtonGroup';
import Title from 'renderer/components/Title';
import Thumbnail from 'renderer/components/Thumbnail';
import BasicFields from 'renderer/components/BasicFields';
import store from 'renderer/redux/store';
import tags from 'renderer/redux/slices/tags';
import { logUserEvent } from 'renderer/util';
import ApplicationSelect from 'renderer/components/ApplicationSelect';
import DetailsTagDropdown from 'renderer/components/tag-components/DetailsTagDropdown';

const ScreenshotConfiguration = () => {
  const [screenshot, _setScreenshot] = useState<ScreenshotElement | null>(null);
  const screenshotRef = useRef(screenshot);
  const setScreenshot = (data: ScreenshotElement | null) => {
    screenshotRef.current = data;
    _setScreenshot(data);
  };
  const editScreenshot = (data: ScreenshotElement | null) => {
    console.log('editScreenshot', data);
```

(a) Screenshot View

Screenshot Configuration

```
const ScreenshotConfiguration = () => {
  const [screenshot, _setScreenshot] = useState<ScreenshotElement | null>(null);
  const screenshotRef = useRef(screenshot);
  const setScreenshot = (data: ScreenshotElement | null) => {
    screenshotRef.current = data;
    _setScreenshot(data);
  };
```

Selected Application Windows

Selected Tags

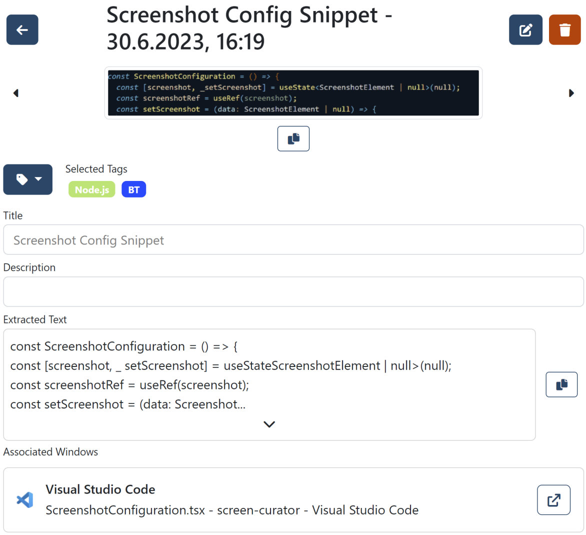
Title: Screenshot Config Snippet

Description: Good example of use of useRef() function

(b) Configuration View

Figure 3.6: Screenshot creation views

When clicking on a screenshot inside the library, one is directed to the detail view of a screenshot, where the title, timestamp, description, tags, extracted text, and associated applications of the screenshot are listed. In this view, you can modify the title, description, and tags of the screenshot. This allows users to correct incorrectly entered values or add new information to already curated items. There are buttons in place to copy the important knowledge containers, namely the screenshot image and the extracted text. Furthermore, the option of deleting or freehand-annotating the screenshot is contained in this view. It is also possible to switch to the screenshot that is displayed next in the library without going back to the library. Users can do this via the arrows next to the screenshot image. This also works properly, even if filters are applied to the library and not all screenshots are shown. These features are implemented to guarantee good usability of the application.



Screenshot Config Snippet - 30.6.2023, 16:19

```
const ScreenshotConfiguration = () => {
  const [screenshot, _setScreenshot] = useState<ScreenshotElement | null>(null);
  const screenshotRef = useRef(screenshot);
  const setScreenshot = (data: ScreenshotElement | null) => {
```

Selected Tags

Title: Screenshot Config Snippet

Description:

Extracted Text:

```
const ScreenshotConfiguration = () => {
const [screenshot, _ setScreenshot] = useStateScreenshotElement | null>(null);
const screenshotRef = useRef(screenshot);
const setScreenshot = (data: Screenshot...
```

Associated Windows

Visual Studio Code
ScreenshotConfiguration.tsx - screen-curator - Visual Studio Code

Figure 3.7: Detail View

Each of these views can be mapped to a step in the information curation life cycle described by Whittaker [Whi11]. On the screenshot- & configuration view, users keep information that is present on their screen. This works also in a multi-monitor setting. Screenshot view windows are opened on every connected screen. The built screenshot collection can be exploited through searches and filters on the library view. Detailed information can be reviewed, altered, and managed through the configuration- and detail view. Therefore, the application is meant to enable the curation of various resources from different cross-application sources and ensure their retrievability. The application library shall result in a flexible and custom repository of knowledge that can be easily searched. The main idea behind the application's interface was to offer the user a simple and organized yet efficient workflow when curating screenshots.

3.3.2 Technical Foundation & Development

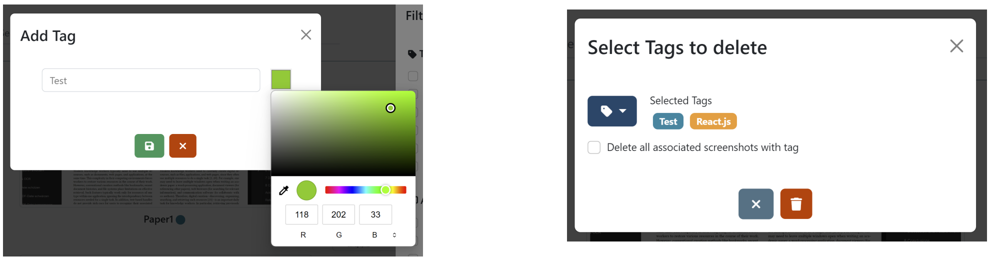
The ScreenCurator application is implemented with Electron.js [Bui]. This framework allows the implementation of cross-platform applications (Windows, MacOS, Linux) using web tools. Electron.js works by rendering the web-specific application code using the embedded Chromium and node.js framework. Furthermore, Electron.js offers platform-specific operating system APIs. The ScreenCurator utilizes these APIs for various features, such as capturing screenshots, creating shortcuts, and exporting data collections. For simplicity reasons, the ScreenCurator is built upon an Electron-React-Boilerplate [Git].

Additionally, the ScreenCurator uses the web frameworks React, Redux, and Bootstrap for the application's frontend. To store curated data in the backend, a persistent in-memory SQLite3 Database was implemented [SQL]. A database file is generated on installation in the local app data folder.

The application was developed in an agile process to optimally support explorative feature discovery. Application features were defined in user stories, which were then saved as Github issues on the project repository [HAS]. These user stories were discussed and prioritized in weekly project meetings.

3.3.3 Screenshot Organization

Instead of a classic folder system, the ScreenCurator deploys a dynamic tagging system. Civan et al. [CJKB08] found that information tagging, compared to a folder organization, is more flexible and can reveal unexpected or forgotten connections. Moreover, it is mentioned that tags better indicate the content of an element than folders do. An advantage of folders, they have found, is that folders hide information and make it easier to process the smaller information packages. The drawback can be removed through the use of broad search and filtering capabilities. According to Civan et al. [CJKB08], the disability to track untagged elements, as well as the lack of possibilities to combine tags can lead to frustration on the user's side. Consequently, it is possible in the ScreenCurator to assign up to 10 tags to a screenshot. An 'untagged' tag is automatically assigned if all tags are removed or none are set. Tags can also be seen as a type of metadata and can easily be filtered or searched for. To profit from visual processing advantages, all tags hold an associated color, which is auto-generated on creation and can be altered by the user. Colored dots in the library indicate which tags are assigned to which screenshots. Users can easily clean up your gallery by deleting tags either individually or along with all the screenshots that are linked to a specific tag.



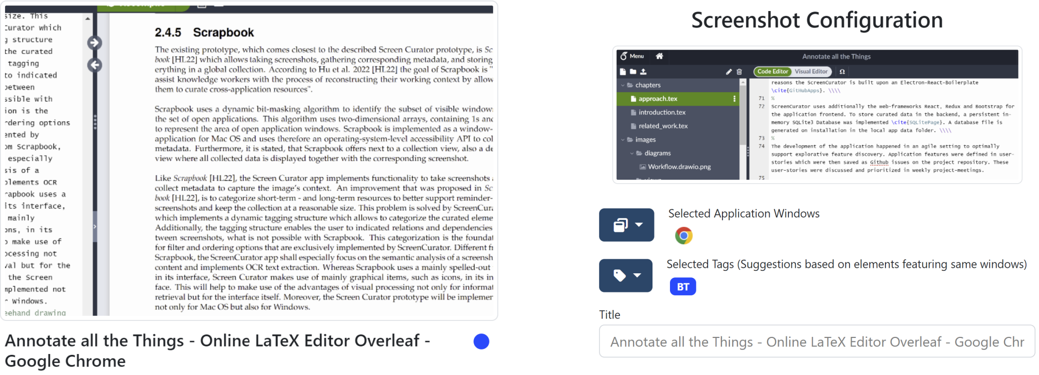
(a) Add Tag

(b) Delete Tags

Figure 3.8: Tag Operations

3.3.4 Metadata

When taking screenshots, the ScreenCurator collects as much metadata about the screen setting as possible to ensure optimal retrievability. The program automatically extracts text elements, gathers information about visible applications in the screenshot, and records the time stamp. Semi-automatically assigned metadata types are tags and titles. Generally, tags must be assigned to the screenshot by the user. Nevertheless, if there exists a curated screenshot featuring the same applications with the same window title, the ScreenCurator automatically suggests the same tags for the new screenshot. The screenshot title is set by default to the window title of the front-most application to offer a meaningful placeholder. Yet, if the title is customized, value can be added because the window title will be saved in the associated applications either way. Moreover, it is more likely users remember a custom title than an application window title.



2.4.5 Scrapbook

The existing prototype, which comes closest to the described Screen Curator prototype, is *Sc book* [HL22] which allows taking screenshots, gathering corresponding metadata, and storing erything in a global collection. According to Hu et al. 2022 [HL22] the goal of Scrapbook is " assist knowledge workers with the process of reconstructing their working context by allow them to curate cross-application resources".

Scrapbook uses a dynamic bit-masking algorithm to identify the subset of visible window the set of open applications. This algorithm uses two-dimensional arrays, containing 1s anc to represent the area of open application windows. Scrapbook is implemented as a window- application for Mac OS and uses therefore an operating-system-level accessibility API to col metadata. Furthermore, it is stated, that Scrapbook offers next to a collection view, also a de view where all collected data is displayed together with the corresponding screenshot.

Like *Scrapbook* [HL22], the Screen Curator app implements functionality to take screenshots collect metadata to capture the image's context. An improvement that was proposed in *Sc book* [HL22], is to categorize short-term - and long-term resources to better support reminder- screenshots and keep the collection at a reasonable size. This problem is solved by ScreenCur which implements a dynamic tagging structure which allows to categorize the curated eleme Additionally, the tagging structure enables the user to indicated relations and dependencies tween screenshots, what is not possible with Scrapbook. This categorization is the foundat for filter and ordering options that are exclusively implemented by ScreenCurator. Different fr Scrapbook, the ScreenCurator app shall especially focus on the semantic analysis of a screensh content and implements OCR text extraction. Whereas Scrapbook uses a mainly spelled-out in its interface, Screen Curator makes use of mainly graphical items, such as icons, in its in face. This will help to make use of the advantages of visual processing not only for informat retrieval but for the interface itself. Moreover, the Screen Curator prototype will be implemer not only for Mac OS but also for Windows.

Annotate all the Things - Online LaTeX Editor Overleaf - Google Chrome

(a) Element previously curated

Screenshot Configuration

```
reasons the ScreenCurator is built upon an Electron-React-Boilerplate
\cite{GitHubApps}. \\\\
%
ScreenCurator uses additionally the web-frameworks React, Redux and Bootstrap for
the application frontend. To store curated data in the backend, a persistent in-
memory SQLite3 Database was implemented \cite{SQLitePage}. A database file is
generated on installation in the local app data folder. \\\\
%
The development of the application happened in an agile setting to optimally
support explorative feature discovery. Application features were defined in user-
stories which were then saved as Github issues on the project repository. These
user-stories were discussed and prioritized in weekly project-meetings.
```

Selected Application Windows

Selected Tags (Suggestions based on elements featuring same windows)
BT

Title
Annotate all the Things - Online LaTeX Editor Overleaf - Google Chr

(b) Suggestion in configuration view

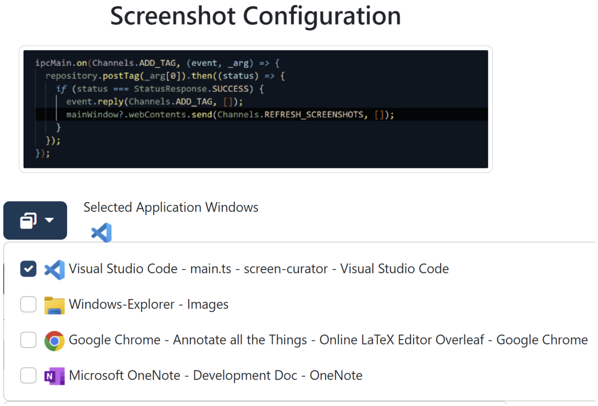
Figure 3.9: Semi-automatically generated metadata

The only metadata that has to be set manually is the description. This type of metadata is thought to serve the user as a field where all additional thoughts and information can be added to the screenshot. The number of not automatically generated or supported fields was minimized on purpose. This shall reduce the workload that is put on the user and consequently minimize the cost of curation. As a result, this shall further encourage the use of the ScreenCurator. The description and the title are metadata types that were also implemented in *Scrapbook* [HL22].

For text extraction, the ScreenCurator deploys tesseract.js, a typescript version of the Tesseract OCR library [Tes]. The text extraction happens in parallel to the extraction of visible windows. To prevent the storage of incorrectly extracted text elements, the user is still able to review and correct the extracted text in the configuration view. However, as soon as the screenshot is saved, it is no longer possible to change the extracted text. This was a design decision to prevent the user from abusing the extracted text field as a container for notes.

To collect all open applications, the ScreenCurator uses the active-win node.js library. This library returns an object for each open window. This object contains the application name, the window title, and the application window borders. The result list is sorted from the front-most window to the back-most window. The ScreenCurator implements a custom algorithm that determines the visibility of an application window based on a set of corner points and the arrangement of other windows. This algorithm is especially efficient for a small number of windows in the screenshot area, which is the expected use case. Since it is quite unrealistic that a user opens more than 50 application windows, there was not a large focus on a low algorithm complexity. In return, the algorithm neither needs down-sampling (reduction of resolution) nor large-scale matrix operations like a comparable algorithm deployed in *Scrapbook* [HL22]. A more detailed description of the algorithm can be found in Appendix D.

To offer the user more flexibility and ensure the possibility of correcting the algorithm result, it is possible to alter the list of visible applications in the configuration view. The idea of collecting open application windows, as well as the user selection option, was originally introduced in *Scrapbook* [HL22] and was seen as a useful feature for the ScreenCurator as well.



Screenshot Configuration

```
ipcMain.on(Channels.ADD_TAG, (event, _arg) => {
  repository.postTag(_arg[0]).then((status) => {
    if (status === StatusResponse.SUCCESS) {
      event.reply(Channels.ADD_TAG, []);
      mainWindow?.webContents.send(Channels.REFRESH_SCREENSHOTS, []);
    }
  });
});
```

Selected Application Windows

☑ Visual Studio Code - main.ts - screen-curator - Visual Studio Code
□ Windows-Explorer - Images
□ Google Chrome - Annotate all the Things - Online LaTeX Editor Overleaf - Google Chrome
□ Microsoft OneNote - Development Doc - OneNote

Figure 3.10: Configuration possibility of visible windows

3.3.5 Search & Filter

To enable intuitive searches, the ScreenCurator features a keyword search. Therefore, all curated screenshots are initially loaded into a search engine, which is implemented using fuse.js. The search engine checks on key-hit if the entered keyword appears in any textual metadata like title or extracted text. Approximate string matching is applied in this case, which means that keywords do not have to be written entirely correctly to be found [Wha].

Furthermore, there are filters implemented for time ranges, tags, and applications. The application and tag filters are also implemented through the search engine. It returns all elements that contain the specific tag ID or application name into the appropriate field. The filters are contained in an off-canvas menu that is triggered by a corresponding button in the toolbar. The application and tag filters offer the user an additional search field to look for a desired filter option. Moreover, the ScreenCurator provides the possibility of saving set filters. These filters can be used to create custom views that are easily selectable. This can help users to quickly switch between different projects and goals. As mentioned in the screenshot organization section, tags could have the disadvantage of hiding too little information and causing an information overload. By providing an easy way to select a restrictive filter, we can overcome this disadvantage and quickly obtain small sets of information.

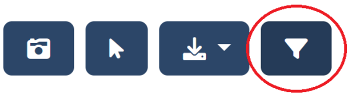


Figure 3.11: Filter Menu Toggle

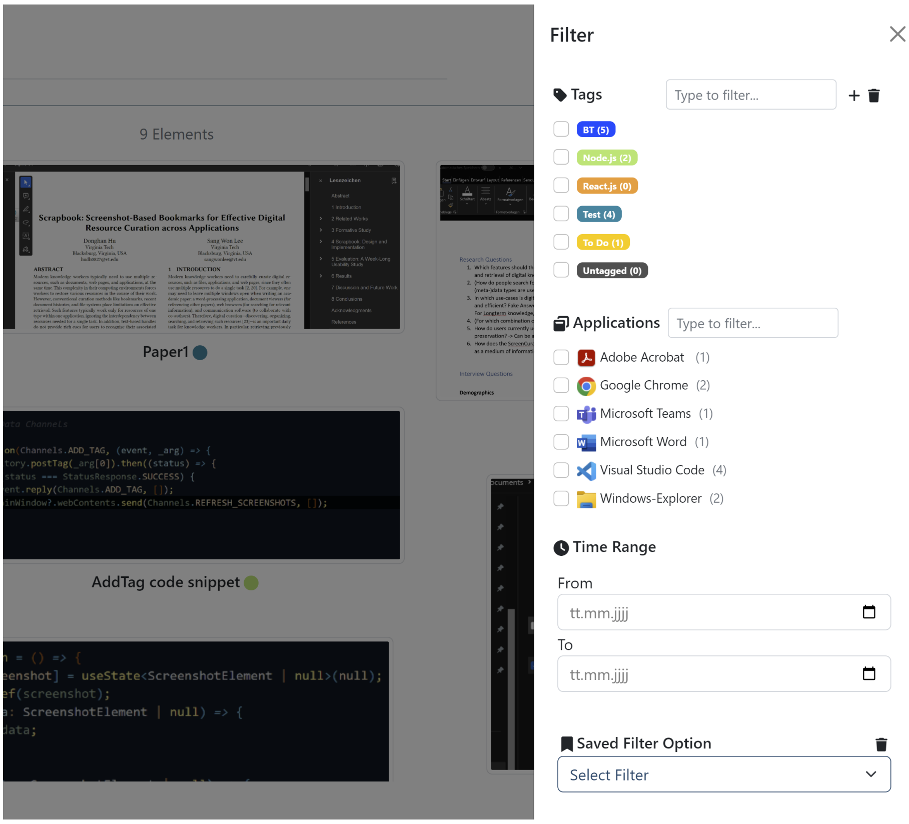


Figure 3.12: Filter Menu

3.3.6 Usability

The ScreenCurator was designed to display information comprehensively while minimizing the amount of hidden information. For example, in the configuration view, selected information is always depicted next to the selection dropdown.

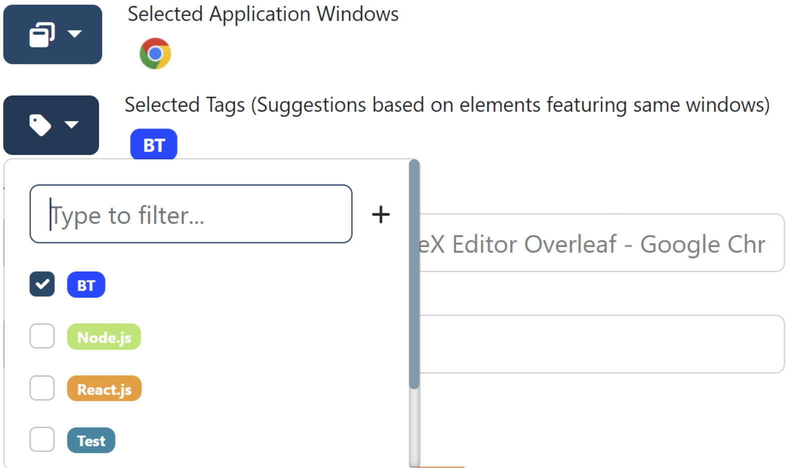

Selected Application Windows
Selected Tags (Suggestions based on elements featuring same windows)
BT
Type to filter...
+
eX Editor Overleaf - Google Chr
☑ BT
□ Node.js
□ React.js
□ Test

Figure 3.13: Information display in configuration view

An exception is the extracted text. This text can get very large and thus consumes much space. For this reason, the text is completely separated in the configuration view and available as an expandable text in the detail view. Typically, action buttons are designed with icons to enhance user understanding and may be specially colored in specific circumstances. For example, all delete buttons are red. To enable fast reusability, the detail view offers copy buttons for the extracted text as well as for the screenshot image itself. Furthermore, the detail view features a reopen button for associated applications. However, this feature is due to technical reasons only available on Windows.

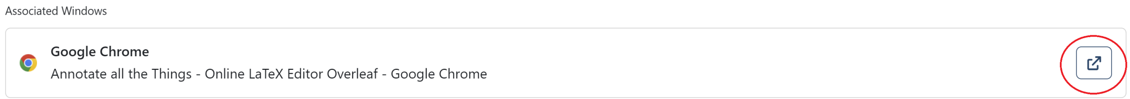
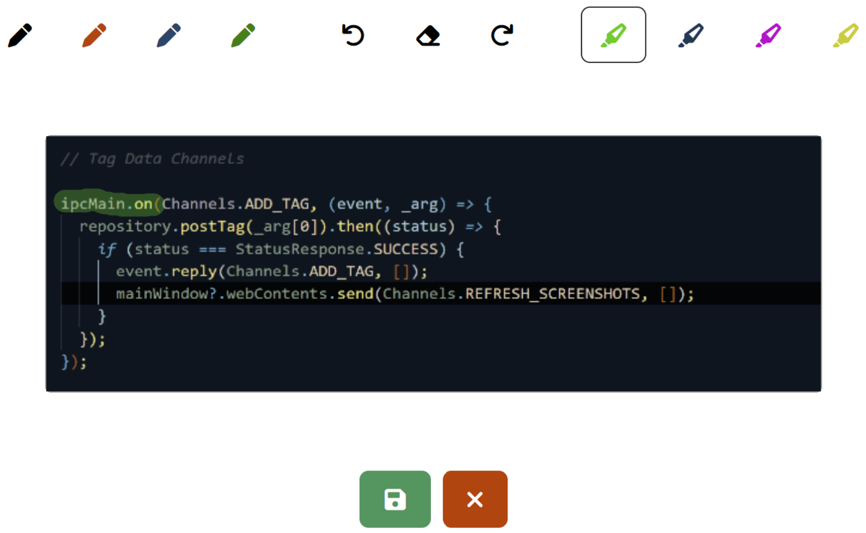

Associated Windows
Google Chrome
Annotate all the Things - Online LaTeX Editor Overleaf - Google Chrome

Figure 3.14: Reopen application

Another feature that shall improve the usability of the ScreenCurator is the freehand-drawing feature that can be opened via detail view. It offers a selection of fine line and marker pens, as well as rubber and forward-/backward operations. This allows the user to highlight important information and augment the image with vital background information.



```
// Tag Data Channels

ipcMain.on(Channels.ADD_TAG, (event, _arg) => {
  repository.postTag(_arg[0]).then((status) => {
    if (status === StatusResponse.SUCCESS) {
      event.reply(Channels.ADD_TAG, []);
      mainWindow?.webContents.send(Channels.REFRESH_SCREENSHOTS, []);
    }
  });
});
```

Figure 3.15: Freehand drawing tool

The library view contains a usability feature that allows the exporting of the collected screenshot. Users can export all screenshots in the library in a folder that is saved in the device download folder. This helps the user to reuse the screenshot if the device is switched or if one decides to not use the application anymore. The library offers, on top of that, an option to select multiple screenshots at once. For the selected screenshots, there exists a delete operation or an option to assign the same tags to all of them. This shall help users to organize a growing screenshot collection.

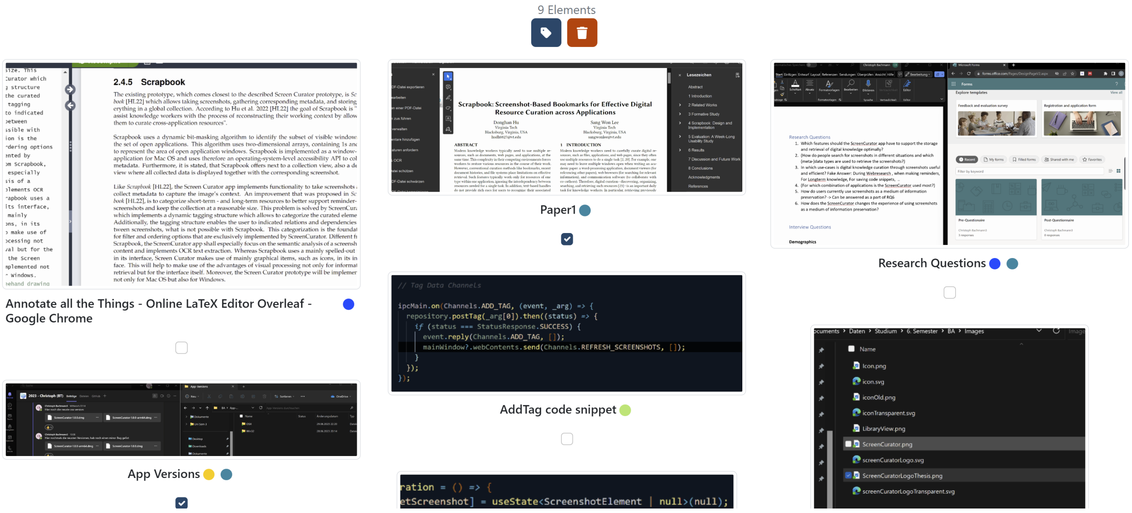

9 Elements
Annotate all the Things - Online LaTeX Editor Overleaf - Google Chrome
Paper1
Research Questions
AddTag code snippet
App Versions

Figure 3.16: Multi selection feature

Chapter 4

Method

In this chapter, it is explained how the ScreenCurator was evaluated and how further insights were generated. A user study was conducted in which seven participants evaluated the application during 10-15 working days. The participants generated three kinds of data that were used for the evaluation: Pre- / post-questionnaire answers, interview answers, and application log data. Participants were all between 23 and 34 years old and had a background in computer science. The data that was generated during the user study provided the basis to answer the formulated research questions.

4.1 Study Procedure

To gather insights about how the ScreenCurator is used, a user study was conducted. In this study, it was of high interest which features of the ScreenCurator are used and how frequently. Moreover, it should be identified in which situations users utilize the ScreenCurator. Furthermore, the participants were instructed that this user study was not about finding minor bugs in the application but testing if the general concept is useful and supportive. The study consists of three different phases and sessions:

- Application Setup Session (max. 20min)
- Trial Phase (2-3 Weeks)
- Interview Session (max. 45min)

In the application setup session, the participants first signed a consent form where they confirmed that they were aware of the procedures contained in the user study. This consent form can be found in Appendix A. Second, they received a version of the ScreenCurator that runs on their operating system. The application was installed together, and participants received a small introduction if they wanted. At the end of this session, the participants were asked to fill out a pre-questionnaire.

The core part of the user study was the trial phase. The participants tested the application in a time range of two to three weeks (10-15 working days). In an optimal case, participants used the ScreenCurator in this time range continuously during work, research, or studying to ensure realistic circumstances of usage.

One of the study goals was to generate quantitative usage data. For this reason, the ScreenCurator implements a logging module that collects data on user interactions. To ensure the participant's data privacy, only types of interactions, like 'applied filter' and item IDs, were logged. So, neither clear-text metadata nor images were logged and, therefore, did not leave the application. To further increase the participant's data security, the entire application works offline and does not deploy any libraries that consume web resources.

At the end of the trial phase, the participants were invited to an interview session. In this session, the participants were asked about their experience with the application. Furthermore, the participants were asked to review the usage data and hand over the log files. Additionally, a post-questionnaire was filled out by the participants in this session.

4.2 Participants

There were 8 participants recruited who participated in the user study. One participant was eliminated from the study due to inactivity. Another participant used the application only for trial purposes. He is, therefore, ignored in real usage questions but is still included in some of the feature- and experience questions. All the participants have been found in the work or university environment of the researcher. The prerequisites that participants in the study had to meet were that they were working on a knowledge-intensive project or writing a thesis. This should ensure that they are inside a knowledge-intensive environment where the ScreenCurator can generate some benefits. All of the participants have a strong background in computer science and work at least part-time in software-related employment. During the study, participants completed a wide range of tasks, including research, project planning, and software development. Six out of seven participants were male, and one participant was female. The age of the participants ranged from 23 to 34, with an average of 27.71 and a standard deviation of 4.92. Participants are referred to as P1 - P7 for further citations.

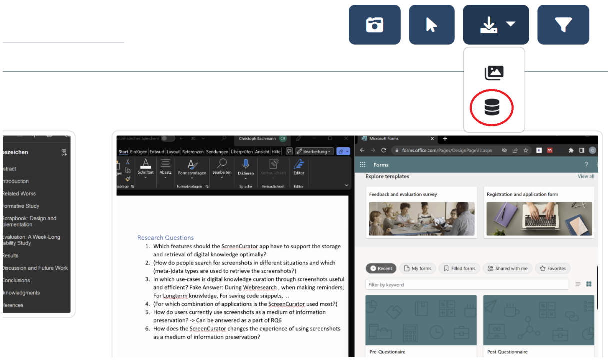
4.3 Data Collection & Evaluation

4.3.1 Pre- & Post-Questionnaire

To generate quantitative data that describes the change in the user experience, users filled out a questionnaire before and after the study. The questionnaire was filled out during the setup session and afterward during the interview phase. The questionnaire was built and managed via Microsoft Forms. The Form contained five Likert-scale questions about the user's general experience with screenshots and their usage. Additionally, three open questions further evaluate how participants use screenshots and for what purpose. In the post-questionnaire, the same five Likert scale questions as in the pre-questionnaire were asked. The reason for not including the open questions in the post-questionnaire was that the answers were not expected to change throughout the study. In contrast, the answers to the Likert scale questions, which express the user's experience while using screenshots, are expected to change. These answers expressed if the experience of using screenshots is improved or worsened by the ScreenCurator. After the study, the results were then statistically analyzed. This offered the quantitative and qualitative data foundation to answer RQ1 & RQ2. Pre- and post-questionnaire can be found in Appendix B.

4.3.2 Usage Data

The log data that had been collected throughout the user study was utilized to verify the qualitative findings from the interview session with quantitative data. Therefore, the participants were asked to share their log data at the beginning of the interview session. Moreover, they were asked to share an export of their database, where personal data was hashed for obfuscation. Participants could easily export these data files by clicking on an according button in the application. The ScreenCurator then created a folder in the participant's download directory and copied the data files into it. The participants had the chance to review the files before handing them over to ensure no critical data was contained.



Research Questions

1. Which features should the ScreenCurator app have to support the storage and retrieval of digital knowledge optimally?
2. (How do people search for screenshots in different situations and which (meta-)data types are used to retrieve the screenshots?)
3. In which use-cases is digital knowledge curation through screenshots useful and efficient? Fake Answer: During Webresearch , when making reminders, For Longterm knowledge, For saving code snippets, ..
4. (For which combination of applications is the ScreenCurator used most?)
5. How do users currently use screenshots as a medium of information preservation? -> Can be answered as a part of RQ6
6. How does the ScreenCurator changes the experience of using screenshots as a medium of information preservation?

Figure 4.1: Log Export

The log files were afterwards analyzed with Python script. This script first extracted the log data from the log file. Then, the log data was cleaned from any unnecessary data and put into a local data format. The script considered special cases for log events to enable maximum flexibility for the logging module. For example, the search log event occurs for each key hit and is compressed to a single event by the script. By default, the script prints the number of events that occurred in the log file and, at the end, the total usage, average, and standard deviation numbers. Additionally, for each log event, specific information attributes were stored. For example, a screenshot access event contains additional information like screenshot ID or if a filter was applied before the access. This additional data built the foundation for multiple functions that analyzed modifications, deletions, and accesses to screenshots. For the hashed database, the number of screenshots and the number of assigned tags and applications are evaluated. Furthermore, this script was able to generate timeline plots based on the extracted information. These processed results were, in turn, used to perform a detailed analysis of the user's interaction with the application.

4.3.3 Semi-Structured Interview

To evaluate the qualitative results of the user study, each participant took part in an interview session. This interview session started with standard questions about the participant's demographics. The rest of the interview was divided into three sections. The initial section involved asking the participants some introductory questions regarding their overall experience. Moreover, it was questioned in which situations the ScreenCurator was used. In the same section of the interview, the user was asked to click on a shuffle button. This button was specifically implemented for the interview session. When clicking on it, the library view shows five random screenshots. The participant will then be asked to explain in which situation and for what purpose these screenshots were made. This introductory section is followed by questions about the participant's opinions about the implemented features. It is questioned which features the participants liked or disliked and which could be added to the application. The last interview section addresses the retrieval of curated elements. It was mainly asked what functionalities participants used in which situations to get back to a curated screenshot. Furthermore, the participants were asked about the time frame in which they plan to reuse the curated information. To classify the participant's feedback, the thematic analysis technique is used. We applied a simplified version of the approach described by Braun et al. [BC06]. The defined codings were discussed and refined in a group of three.

Chapter 5

Results

In the following chapter, the results of the user study are presented. Participants had an overall good experience while using the ScreenCurator. They stated that it improved the process of taking and retrieving screenshots. Participants had various individual use cases, which could be grouped on a high level into long-term backups and short-term to-dos. The participant's favorite features were as different as their use cases. Furthermore, they had multiple ideas for features to add. The most important ones are more automation for the capturing process and additional structuring options for captured items.

5.1 Screenshot Curation Experience (RQ1)

The first research question was intended to examine if and how the ScreenCurator changes the participant's experience of working with screenshots. Before the user study, participants mentioned that the structuring and retrieval of screenshots is laborious. After the study, it was visible from the participant's feedback that the application brought some overall improvements to the participant's experience.

Screenshots present an excellent medium for information capturing in terms of expressiveness and intuition. Nevertheless, they are cumbersome to manage and retrieve. Before the study, six out of seven participants indicated in their answers the cumbersome search and retrieving process of previously made screenshots. P5 stated:

> "Images usually need to be stored in a structured way. It is difficult to find the information again because you can't search the content on the image."

To further elaborate, three participants mentioned that one cannot add further information or metadata to screenshots. This leads to a very limited set of options to search for a screenshot. One must count on a meaningful file title or still know the estimated time range when a screenshot was taken.

On the other hand, participants were also asked about the benefits they see in using screenshots. Participants mentioned already having used screenshots to capture visual information or resources that were otherwise hard to curate. Most of the participants appreciated the concepts of high expressiveness and simple collection. Three participants answered the question about the advantages of screenshots with the metaphor "A picture says more than a thousand words."

After the user study, participants were much more optimistic about managing and retrieving screenshots. Participants were asked if they consider the organization of screenshots laborious. While five of seven participants were neutral or agreed before the study, six out of seven disagreed after the user study. Therefore, no participant found it laborious to manage and organize screenshots with the ScreenCurator, which is a great improvement. The same can be seen for the retrieval of screenshots. Before the study, four out of seven participants disagreed when asked if they could easily retrieve screenshots. After the user study, all seven participants agreed that this would now be effortless. Consequently, the ScreenCurator cancels out two main disadvantages of working with screenshots. Moreover, before the study, three of seven participants stated, to not structure their screenshots for further retrieval. After the user study, all participants were neutral or agreed to structure their screenshots. This means that participants structure their screenshots more often with the ScreenCurator.

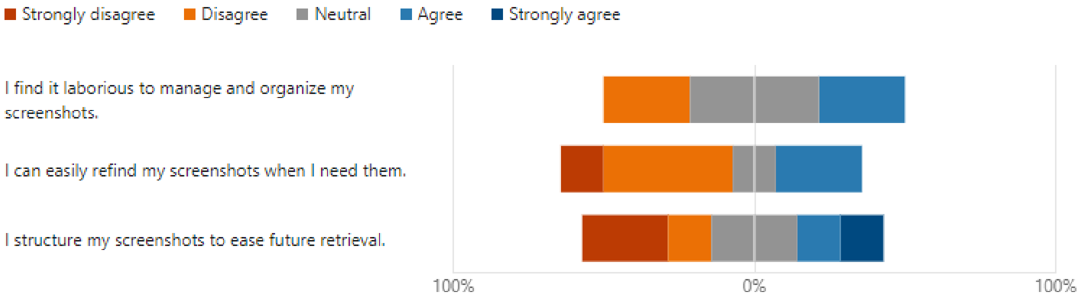


Figure 5.1: Answers before the user study

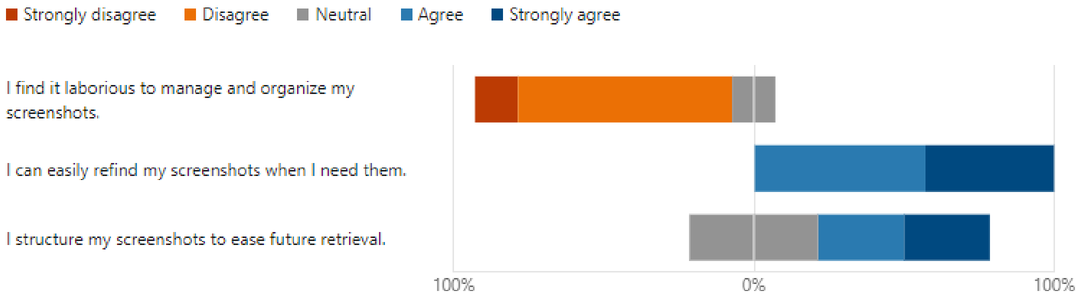


Figure 5.2: Answers after the user study

In addition, all seven participants reported having a positive experience when using the ScreenCurator. Three out of the participants mentioned that they found it especially nice they could quickly make a screenshot and later know exactly where and how to retrieve it. P5, for example, stated:

> “I thought it’s nice that you have an overview of all the screenshots and that you can always go back to screenshots you have previously made.“

P3 even announces that the application could change his way of taking notes:

> “Previously, I did not really work much with screenshots (...) but this tool introduced me to a new way, and I think this might be helpful for specific use cases in the future.“

5.2 Screenshot Curation Usage and Purpose (RQ2)

The second research question targets different use cases for a screenshot information curation application. The participant's answers indicate that especially the archiving and backing up of information seems to be a popular application of the ScreenCurator. Besides that, the concrete range of use differed a lot between the participants. A similarity of usage is the common curation of information captured in browsers.

5.2.1 Active Application Usage, but Not All Features Tested

We see an active usage of the application's main functionalities during the user study but little to no usage of extended retrieval and modification functionalities. During the study, six participants created 81 screenshots and eleven tags, on average 13.5 screenshots and 1.8 tags per person, with a standard deviation of 5.1 and 1.2. The high standard deviation can be explained by the fact that there was a large range of different levels of activity. Participants captured between 4 and 21 screenshots and created 0 to 3 tags. After the screenshots had been curated, fourteen modifications were made to the titles, descriptions, and tags of twelve screenshots. Drawings were also included as modifications. Eleven tag filters were applied, and 17 searches were carried out. Overall, 73 accesses happened on 44 screenshots. This is an average of 12.2 accesses on 7.3 screenshots per person, with a standard deviation of 7.0 and 3.7. The accesses ranged between 6 and 23 and accessed screenshots between 4 and 15. Participants deleted 16 of the 81 screenshots during the user study and had cumulated 65 screenshots at the end. Nonetheless, multiple features were not used by the participants. The participants did not save or reuse any filters, nor did they apply time range or application filters.

When looking at the timeline diagram of curated screenshots (Figure 5.3), one can see that many screenshots were taken at the very beginning of the user study. The same is true for screenshot accesses (Figure 5.4). These operations could be made due to the desire of the participants to explore all the application's functionality. In Figure 5.3, we see that P3 and P2 took screenshots rather regularly in the first half of the user study and less in the second half. For P1 and P6, it is exactly the other way around. P4 and P5 took screenshots sporadically. In Figure 5.4, we see that, especially P2 and P1, continuously accessed screenshots throughout the user study. The rest of the participants accessed screenshots rather sporadically. Nevertheless, it must be taken into account that the analyzed screenshot access data only shows how often users went to the detail view of a screenshot. Despite accessing the detail view, it was also possible for participants to reuse curated knowledge by looking at screenshots in the library.

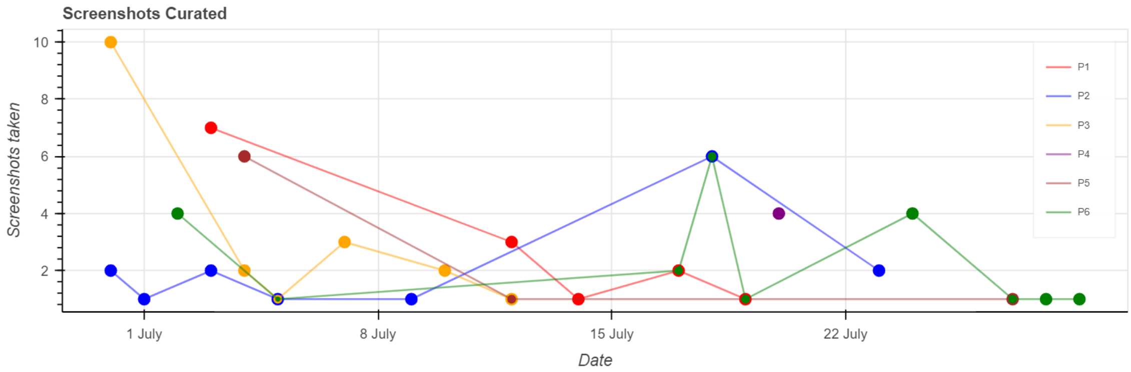


Figure 5.3: Timeline Diagram Curated Screenshots

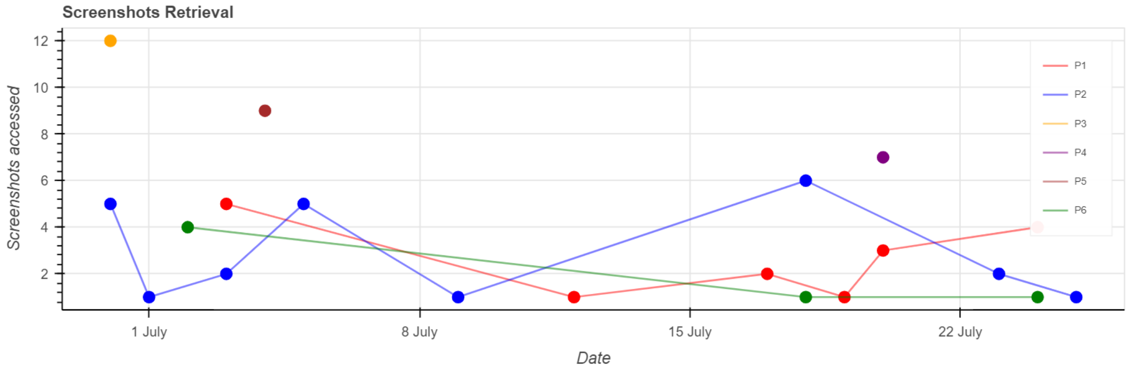


Figure 5.4: Timeline Diagram Retrieved Screenshots

5.2.2 Users Captured Only Single Applications, Primarily Browsers

Most participants captured a great part of their screenshots in a web environment but also collected screenshots of non-browser applications. Therefore, the combination of web- and local resources seems to be a popular use case for the ScreenCurator. P1 states:

> “Especially if I have to search for it again somewhere in the future, I have everything in one place. It doesn’t matter where I’ve saved it. So everything is in the tool.“

None of the participants had screenshots that featured more than one application. The screenshot collection of five of six participants consisted of a great part of screenshots made in a web browser. Out of the 65 screenshots that remained at the end of the study, 44 of them showed a web browser. Therefore, more than 2/3 of the screenshots were made inside a browser. Nevertheless, five of six participants also had screenshots of other applications. Four out of these five participants captured communication platforms, like Microsoft Teams and email clients. Two out of these five participants captured screenshots in a Coding Environment, like Visual Studio Code. Furthermore, it must be mentioned that all applications were captured on separate screenshots.

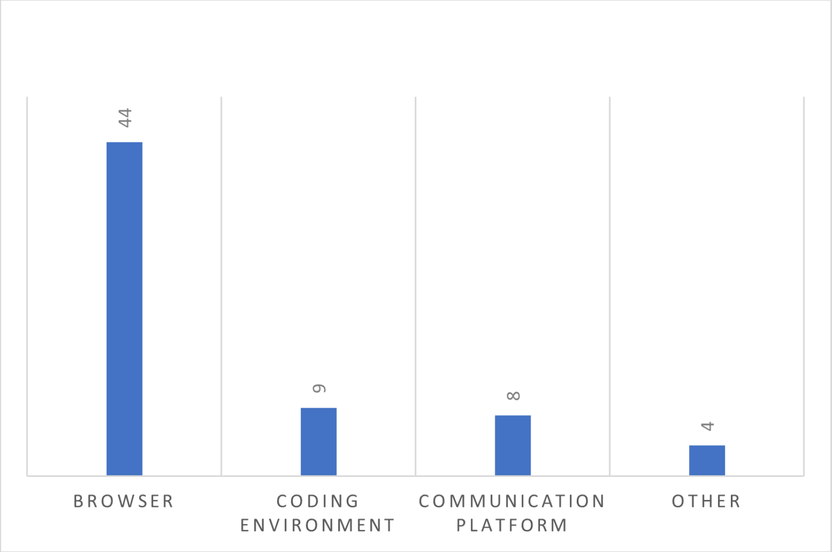


Figure 5.5: Captured applications

5.2.3 Long-term- & Short-Term Use Cases Identified

Users primarily utilized ScreenCurator for use cases with the goal of long-term curation. Nonetheless, short-term use cases like to-dos could also be found among the screenshots the participants captured. Therefore, a co-existence of these elements could be found in the user study.

Backups, archiving, and information preservation were the most named use cases for the ScreenCurator. All of these use cases have a long-term curation scope. Five out of six participants stated to have used or will use the ScreenCurator for these purposes. This is underlined by the fact that all seven participants would keep screenshots containing general information for more than a year. Moreover, the logs show that the six participants took 81 screenshots in total but only accessed 44 of them. 20 of the 44 screenshots were accessed multiple times. So, almost half of the screenshots were never accessed during the study. Additionally, screenshot accesses for application testing purposes should be taken into account.

The detailed use cases of the five participants who kept screenshots for backup and archiving purposes varied. Popular use cases were the curation of problem solutions and configurations. Three participants used the ScreenCurator to curate problem solutions. They saved solutions to problems they encountered so that they could reuse them if they faced the same issue in the future. P3, for example, stated:

> "[I use the ScreenCurator for] storing for later reference, but mostly like backups, reference and whenever a situation occurs again I can reference it."

This use case represents the core idea of information curation due to the kept, organized, and planned reuse of knowledge. Two out of these five participants curated configurations. They used the ScreenCurator to curate configurations because they were in a visual format, hard to copy, or wanted to highlight something. Problem solutions, as well as configurations, came from different contexts like online forums, AWS cloud, and code environments. Three of these five participants used the application to preserve project resources like images or logos and have a backup of them.

Apart from long-term information resources, participants also used the application to capture short-lived reminders or to-dos. Nevertheless, it must be mentioned that participants did not properly sort out outdated short-term assets. Two out of six participants used the application to manage short-term reminders and to-dos. During the interview session, they had still short-lived elements in their library, which were outdated. The hypothesis of the lack of cleanups is supported by the fact that all logged screenshot deletions are expected to be tests since all of them were made in a very short time range after the capturing. Additionally, most deletions happened on the first day of the user study.

Both Participants had the use case of bookmarking important emails. P2 also used the application for project planning. P5 made screenshots to extract important sections from emails and have them available as tasks. P2 took screenshots of emails as a reminder to review interesting newsletters. One of six participants (P2) used the ScreenCurator for the planning of personal projects like holidays. They further mentioned that the saved screenshots enabled sharing fixed plans with friends.

5.3 Application Features (RQ3)

The last research question is stated to evaluate which of the implemented features are useful and what features still need to be added. Participants had various ideas of what features were especially useful and which ones could be added. Nonetheless, most participants agreed that the high-level topic of automated support could be improved and extended. The answers to the questions about the structuring options of the ScreenCurator implied that personal preferences are crucial for the effectiveness of a structuring system. A usability and augmentation feature that multiple participants liked was the freehand drawing tool.

5.3.1 Demand for More Workflow Automation

Participants did not invest much work when it came to custom assignment and augmentation of metadata. This implies the benefits of further automation. High-quality metadata is crucial to enable an effective search and filter functionality. Therefore, automated metadata collection was a high-level topic where many participants saw the potential to enhance the application. This is supported by the fact that five out of six participants had screenshots on which no tags were set. The log data shows that only 19 of the 65 screenshots were tagged. Two out of the five participants who had untagged screenshots mentioned that the effort of creating a tag was too high for them. When asked why they did not tag his screenshots, P3 answered:

> “I think just because it was faster to not do it and I was like sort of relying on the interface(...).“

Five out of seven participants mentioned that they would have liked more automated support when it comes to the tagging of screenshots. All of these five participants would have liked an automatic tag suggestion for a screenshot. This suggestion can include existing as well as new tags. One of these five participants (P1) would have liked to see a warning if they tried to store a screenshot without setting tags.

A new feature in the automation domain that was proposed by two of the seven participants was the automated extraction of URLs from captured browser windows. This proposal is further supported by the fact that two of the participants wanted to copy and paste URLs from browser windows into the description. Unfortunately, they thereby discovered a MacOS-specific bug that prevents the copying and pasting of information into application text fields.

Participants liked the automation that was already implemented in the ScreenCurator, except for a few quality issues. One participant mentioned that they were positively surprised about the tag suggestion for screenshots with the same window titles. Another participant especially liked the default screenshot titles, which are the titles of the front-most application.

Nevertheless, participants saw room for improvement in the quality of the existing automation, especially for the extracted text and generated titles. Five out of seven participants disliked the quality of the text extraction for multiple reasons. P2 states that the visual grouping of text elements in an image gets lost completely:

> "This made it hard to use the extracted text because there was no way of logically correlating the text to the screenshot."

Moreover, P4 noticed that the text extraction only works for English text elements and that German text was extracted with very low accuracy. P1 mentions that the quality of the text extraction especially needs to be improved to enable the direct reuse of the extracted text. They stated:

> "I've experienced that the text extraction is not so good as I had a lot of screenshots where I actually had text that I wanted to reuse, but I had to fix it somehow all the time."

Out of the seven participants, two stated that the default title was sometimes not precise enough. P1 mentioned that sometimes useless browser window titles were selected as default titles. These described the overall web pages but not the information that was captured.

5.3.2 Personal Preferences Vary for Library Structure

The tagging concept, as a feature on its own, split the participants into two groups. Despite this, the participants did not make extended use of the tagging feature. Participants were divided almost in half with their preferences about the structuring of curated elements. Four out of seven participants liked the approach of structuring screenshots by tagging them. P1, for example, liked the dynamic views that could be created by the tags:

> "I think with the tagging, it's like a catalog function, where you see the data from different perspectives."

P3 liked the 'fresh' approach:

> "I think this is a fresh approach because this gets you an overview, and you see directly what screenshots you have."

The other three participants would have preferred a folder structure to organize their screenshots. P2 mentions that they would have liked folders because they offer the possibility to drag and drop elements. Additionally, they like the hierarchical navigation that folders enable. P6 mentions that they are very much used to using folders:

> "I really don't know how to use tags at all. I'm already a folders guy (...), and the workflow with tags just doesn't come naturally to me."

However, no matter what opinion the participants stated about the tags, it must be mentioned that tags were overall not used very often. As stated in the automation subsection, only 19 out of 65 collected screenshots were tagged. Four out of six participants mentioned that their collection or their sets of categorizable screenshots were too small for the effort of creating a tag. P3 states:

> “(...)I’m going to find [a screenshot] easily without tagging it. But of course, if [the collection] is growing and I’m taking more and more screenshots, then the tagging system obviously would make much more sense(...).“

5.3.3 Search Strategy Depends on Library Size

Participants stated that, during the user study, they mostly scrolled through their library to search for a specific element. Nonetheless, the library size seems to be crucial for the question of how screenshots are retrieved. The keyword search was, in total, only applied in 10 out of 73 screenshot accesses. This means that ten screenshots were searched with the keyword search and were shown in the library due to a match in the metadata before they were accessed. In total, only 18 keyword searches were performed over the whole user study. The tag filter was applied in eight cases when a screenshot was accessed and was used eleven times by the six participants. 55 screenshot access happened when no filter or keyword search was applied, which means that they were found by scrolling through the library. So, only in roughly 25% of the screenshot accesses, search, or filter functionality was used, which is quite a little usage. Furthermore, no application and time-range filters were used according to the application log data. The non-frequent usage of search- and filter options can be explained by the fact that four out of six participants mentioned that they had too small libraries to benefit from this functionality. P6 states:

> “I was just scrolling. But I mean, I used it on the big screen at home, and most of the time, I was able to see all the screenshots at once. So, searching for them was not that hard“

During the interview session, the remaining two participants stated that they had not yet actively retrieved any curated elements.

Participants were also asked if they could imagine how they would have searched a larger library. Five of seven participants mentioned that they would have used the keyword search in a larger library to retrieve curated elements. Also, five of seven participants mentioned that they would have used the filter system for tags, applications, or time ranges to retrieve screenshots in a larger library. Additionally, it must be mentioned that none of the seven participants would only use scrolling in a larger library.

5.3.4 User-Friendly Interface, but Improvements of Individual Elements Needed

Besides the fact that participants had an overall good impression of the application interface, it has become evident that certain elements are unintuitive. Four out of seven participants specifically mentioned that they liked the overall simple, clean, and intuitive interface. P6 states:

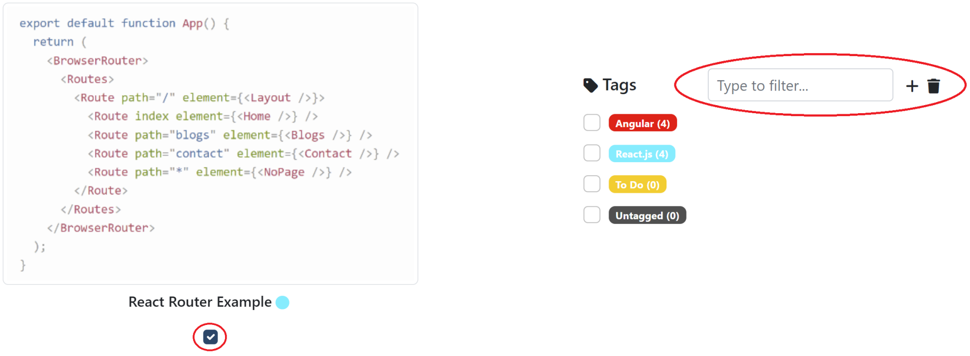
> “Overall, I really liked the application you provided. I feel like the interface and usability are actually really great(...).“

P3 mentions that the interface was one of his favorite features of the application:

> “(...) I really like the interface. I think it’s very simple, so it’s easy to use.“

Nevertheless, three participants revealed that they encountered some unintuitive elements and that the interface needs some improvements on specific elements. Specifically, their visibility and room for unintentional interpretation need to be improved. P4, for example, stated that the multi-screenshot-selection interface could be improved with better visibility of the screenshot selection.

Namely, they proposed that a screenshot can be selected by clicking on an image instead of a checkbox beneath the screenshot. P3, in turn, mentions that the creation of a tag would be more intuitive if it could just be added after typing it into the tag filter bar in the filter menu instead of opening a modal.



```
export default function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Layout />}>
          <Route index element={<Home />} />
          <Route path="blogs" element={<Blogs />} />
          <Route path="contact" element={<Contact />} />
          <Route path="*" element={<NoPage />} />
        </Route>
      </Routes>
    </BrowserRouter>
  );
}
```

React Router Example

☑

(a) Multi-Screenshot-Selection Checkbox

(b) Tag Filter Bar

Figure 5.6: Possible Interface Improvements

5.3.5 Participants Had Divergent Opinions on What Features Are Essential

The features that the participants liked best, as stated in the interview, varied greatly from person to person. The same applies to the interview question about which features participants think should be added. Participants liked data augmentation, retrieval, automation as well as sharing features. The Freehand drawing editor, the keyword search, and the concept of text extraction were also popular. A preview and more structuring options were among the most wished-for features.

Participants liked the ScreenCurator's data augmentation features, including the freehand drawing tool and the description field. Nevertheless, the usability of both features could be enhanced. Three out of seven participants counted the freehand drawing tool as one of their favorites. It was especially liked for its simplistic interface and the possibility to quickly highlight important sections. Nonetheless, two of the three participants would have wished for draggable elements like arrows or circles that could be added to an image. Furthermore, text fields that could be added to an image were wished for by P7. Despite the positive feedback from the interview section, the application log data shows that only five images were annotated by four participants. So, the usage of the freehand-drawing tool during the user study was yet quite limited. Participant P4 implied during the interview that the freehand drawing tool is only of use if a screenshot has a certain complexity and is not self-explaining. Two participants enjoyed the description field as a data augmentation feature. It was especially liked due to its functionality as a container for any additional thoughts and information. Only P4 mentioned that a multi-line field with optional mark-down support would be vital to enable efficient usage.

The retrieval features, including the keyword search and the filters, were generally appreciated by participants. Three participants especially liked the keyword search. P6 states that they found it particularly impressive that the keyword search targets the text inside of images. Furthermore, P1 states the keyword search worked efficiently for retrievals. The filter options counted among the favorite features of one of the seven participants. P7 liked this feature for its simplicity and efficiency.

As previously stated, the participants generally appreciated the current automation features of the application, although there is room for improvement in terms of quality. The automated metadata suggestions, including the default title and the tag suggestion (based on the same window titles), were especially liked by one participant each. Three of the seven participants explicitly liked the overall concept of automated text extraction. Nevertheless, all participants agreed that the current implementation of the text extraction needs to improve.

Nearly all participants had different ideas about which features could be added. Two participants suggested adding a preview feature in addition to the detail view. This preview shall simply enlarge the image without the need to go to the detail view. P2 requested a feature that allows users to share screenshots collaboratively. They state:

> "If I share the screenshots, I just have to copy them if I want to share a selection of screenshots, it would have maybe been cool to have something there (...)."

This feature request is supported by the fact that three out of seven participants mentioned in the pre-questionnaire that they have used screenshots to share information. Despite it, P2 mentioned having shared screenshots made with the ScreenCurator simply by using the 'copy image'-button and pasting it into a communication application. That is why they specifically liked the copy option for single screenshots. A related feature was asked for by P3, who would have enjoyed it if they could import previously made screenshots into the screenshot collection. A feature regarding the structuring was proposed by P4, who would have specifically liked an option to join two or more screenshots that belong strictly together. P4 mentioned a case where they wanted to capture information that was located at the top and the bottom of a website. As a result, they had two choices - either zoom out until the text became difficult to read or take two screenshots to capture all the information. In the second case, they had to create a new tag only to mark the link between the two screenshots. Therefore, they thought joining two screenshots instead of a designated tag would be more helpful and would make the access of captured information easier.

Chapter 6

Discussion

Our application improves the overall experience of working with screenshots. Nevertheless, enhancements in usability, structuring options, and automation can be done to optimally support knowledge workers. Furthermore, users would benefit from a collaborative feature to efficiently share acquired knowledge with colleagues. We also found that participants' curated resources can be grouped into long-term- and short-term assets. Nonetheless, participants used the ScreenCurator for many different individual use cases, which is why it is vital to offer them as much flexibility in the curation process as possible. This is important to support all the potential use cases equally.

6.1 Improved Screenshot Curation Experience

The ScreenCurator improves the experience of managing and retrieving screenshots by counteracting their disadvantages and exploiting their advantages. Regarding the disadvantages of screenshots, which the participants mentioned before the study, it became clear why the ScreenCurator improves their overall experience. Participants explicitly wrote down the cumbersome search and retrieval of screenshots as one of the main disadvantages. One of the core concepts of the ScreenCurator is to ensure good searchability and, therefore, good retrievability of screenshots. The ScreenCurator enables this through the combination of metadata assignment as well as extended search and filter options. Furthermore, the advantages of screenshots the participants named, such as expressiveness and simple collection, are still preserved in the ScreenCurator. The library view takes advantage of the screenshot's expressiveness by displaying all screenshots and, therefore, allowing users to visually identify searched items. The snipping tool is designed to preserve the advantage of simple but precise information collecting. As a result, the experience of retrieving a screenshot seems to be improved by the ScreenCurator when considering the feedback provided by participants. Nonetheless, it must be mentioned that the analyzed questions explicitly target the structuring and retrievability of screenshots.

The ScreenCurator extended the common possibility to assign and augment further information to screenshots, even if there is still potential to add options. The lack of possibility to add more information or metadata to a screenshot was another disadvantage participants saw before the study. The ScreenCurator resolved the issue to some extent by introducing various metadata fields that allow users to include different types of information. Specifically, the description field seemed to allow users to flexibly add information and was popular among participants. The tagging system also allowed the participants to add further information to an existing topic. Nevertheless, the possibility to add and combine further knowledge can be extended. A possible solution would be to enable the joining of closely related screenshots. This would allow users to later combine single units of information that are not of use on their own.

6.2 More Combination Options for Single Application Screenshots

It appears that participants only captured single applications instead of cross-application information, which could be the result of too high capturing costs. More structuring options could lower capturing costs of multi-application settings. The high collection costs may arise due to the difficulty of arranging windows side by side in preparation for a screenshot. One possibility for users to combine information from multiple applications would be the previously mentioned possible feature of joining single screenshots. Users would be able to combine individual units of information without moving application windows in place or abusing the tagging system to combine two elements. Therefore, the cost of window arrangments and extra tag creations could become obsolete. Nonetheless, it remains unclear how prevalent the use case of cross-application information capturing is compared to the capturing of single-application information. However, since the study revealed that users have many different use cases, this one should also be considered to offer more flexibility.

6.3 Offer Flexibility to Enable Various Low-Level Use Cases

When examining the sources of information used by participants, it became clear that browsers were the most widely used application. However, flexibility was still a key consideration due to the variety of use cases and sources involved. Concrete use cases were as different as the tasks that participants completed during the user study. They ranged from holiday planning to saving backups of cloud resource configurations. Therefore, it is crucial for the application to offer users as much flexibility as possible and support users in all different curation use cases. This flexibility shall especially be offered in what metadata is assigned and how items are structured in the collection.

A significant portion of the screenshots taken by participants feature a web environment. This is not surprising considering that software developers often rely on the internet to access fundamental technical and external product information [Fre15]. Furthermore, many commonly used applications like Microsoft Office and cloud management consoles are available as web applications. Still, most participants also curated screenshots from other applications like coding environments. This underlines the importance of a central repository to store knowledge from all kinds of sources. It is important to offer the flexibility to add information from various sources beyond the internet for a comprehensive approach to information curation. Although participants did not save cross-application data in single screenshot items, the majority of them stored data that was captured from various sources in multiple screenshots in their collections. This, in turn, emphasizes again the importance of the cross-application-curation concept.

6.4 Limited Evaluation of Long-Term High-Level Use Cases

In the user study, two high-level usage patterns could be identified. On one hand, screenshots are used for long-term archiving and backup purposes, on the other hand, they are used for short-term reminders and to-dos. This matches the findings of Hu et al. [HL22], which mention in their future work section that a collection view (library) tempts users to store short-term reminders next to long-term resources. These use cases can now be cleanly separated by using the built-in to-do tag. To examine the balance between short-term and long-term resources thoroughly, a user study spanning a longer time period would be necessary. The same conclusion is valid for the research on the actual benefits provided by curated long-term resources. Since almost half of the screenshots were never accessed during the study, there is no indication of large-scale reuse of captured resources over the time range of the user study. Furthermore, accessed elements most certainly included test access and retrieval of short-term reminders. Since the user study only ranged over 10 to 15 working days, the period was too short for participants to actively reuse curated long-term resources. This was stated or implied by multiple participants. P6, for example, mentioned:

> "(...)I'll come back later when I forgot what the page was called, and I can just barely remember that I took a screenshot of it. That would be really helpful, I think(...)".

Most of the participants had curated knowledge still in mind and stated to assumably reuse the resources in a much more long-term scope.

6.5 Collaborative Feature as a Useful Enhancement

The popular use case of collaboratively sharing information with screenshots could further simplify and extend the reuse of knowledge. This use case did not get much attention in the development of the ScreenCurator. The only features that could be used for this purpose are limited to copying and exporting images. Nevertheless, multiple participants stated before the study to use screenshots often to share information with co-workers and colleagues. To support the collaborative use case of the ScreenCurator, many more sharing and interaction features would be required. This could include an option to share single screenshots or whole tag groups with other users. Another option would be to facilitate team libraries, which are stored as shared resources. Furthermore, a comment option could help teams to collectively add knowledge to a resource. However, all collaborative features, in turn, would require a web-hosted backend for the application to enable remote sharing and communication. As a result, the application would become much more complex. Moreover, the existing concepts to ensure users' data privacy and -security would need to be revised.

6.6 More Automation Is Needed to Further Decrease Collection Costs

Intelligent tag suggestions and automated URL extraction for web resources could efficiently support the assignment of metadata for screenshots. The participant's feedback indicates that more automation in the process of curating screenshots would be much appreciated. Potential can be especially seen in the area of automatic tag assignment and grouping of resources. The library view can quickly get unorganized when items are not stored in a structured way. Consequently, curated elements could be harder to find in the collection, which endangers the concept of screenshot retrievability. Improved automatic tag assignment could be very useful, considering that participants rarely tagged their screenshots. Furthermore, it could decrease the cost of capturing and curating pieces of information. In turn, the curation of knowledge would become overall more efficient. This is emphasized by interview answers in which participants mentioned the high effort of creating and maintaining tags, especially in an initially small collection. Nonetheless, the specific implementation would probably require enhanced artificial intelligence and add a lot of complexity to the application.

The second automation feature that could be of great value for the ScreenCurator would be automatic URL capturing for browser screenshots. Particularly when looking at the high number of screenshots that were taken in a browser, the usefulness of this feature is underlined. Users would be provided with a direct link to a screenshot source and would be able to reuse further information from the same source. The implementation of such a feature could happen via a browser plugin that can communicate with the ScreenCurator.

6.7 Quality of Automation and Customizability Essential for Usability

Automated metadata generation must be of high quality and always customizable to deliver actual benefits. Inaccurately generated metadata can lead to frustration among the users, as the text extraction showed in the user study. Many participants liked the concept of text extraction, but still, it was among the most disliked features due to its quality. The reuse of the extracted text was often not possible due to ignorance of visual text structures and wrong character identifications. Although it is often impossible to eliminate the possibility of incorrect suggestions, we should strive to minimize it as much as possible for optimal benefit. Therefore, further automation features should only be implemented if they are fully developed and tested. This would be costly, especially for AI-supported automation, like the creation of new tags. Automated metadata extraction that could be done via system APIs like URL extraction would be, in turn, easier to implement in acceptable quality. Despite the degree and the quality of automation, it remains crucial that users can customize all automatically collected metadata. This ensures sufficient precision in the case of especially important curation elements.

6.8 Usability as Tradeoff Between Functionality and Simplicity

To provide optimal application usability, it is important to enable the previously mentioned flexibility for various use cases but still keep the interface simple and intuitive. Participants overall had a good experience using the ScreenCurator and appreciated the simplistic design. The application's limited and straightforward features enabled participants to navigate it quickly and easily. As a result, the usage of the application felt intuitive to participants, which was one of the core concepts. However, to accommodate the flexibility for different individual use cases, a multitude of small features are necessary. For example, if we implemented all wished-for features, like the preview of screenshots and the import functionality. This will provide the flexibility to access screenshots more easily and allow the use case of curating screenshots captured before the installation of the ScreenCurator. However, this will let the application's feature set grow very fast. As a result, the application will become more complex and, therefore, less intuitive.

A feature that is illustrative of this tradeoff is the freehand drawing tool. This feature was popular among the participants, especially due to its simplicity. Nonetheless, further features, like text areas and insertable shapes, were proposed by the participants to be able to augment screenshots in a more detailed way. These features could cause the interface to get more packed, and users would, in some cases, need more time to familiarize themselves with the functionality. This, in turn, would decrease the application's intuitiveness. This is why it should be carefully evaluated which features are added.

Despite the size of the feature set, for each implemented feature, it should be clear to the user what the feature does and how it works. Therefore, it is important to fix all unintuitive elements that were found in the user study, including the tag creation- and the multi-select interface. Furthermore, the application could benefit from proper beta testing to identify all unclear interface elements.

6.9 Threads to Validity

The external validity is threatened by the fact that the number of study participants was rather low. The user study result provides valuable preliminary insights into the application usage and the experience it provides. Moreover, the concept of the application could be preliminarily evaluated for its usefulness. Participants used the application both in a university environment and an industry context. Nonetheless, the study results should be replicated in a second user study, including a larger number of participants. Another limitation of our user study was the short study period. The actual benefits of curated long-term resources could not be accurately evaluated. Moreover, the participant's collections were often too small for the search and filter options to be used in a realistic setting. Nevertheless, short-term and some long-term resources were reused during the study. Additionally, much of the application's functionality and features could be tested by the participants anyway. They could state their initial experience with the concept of screenshot-based information curation.

The study's internal validity could be at risk since the researcher only recruited participants from their social circle. They knew they would be interviewed, and data about their interactions would be collected. It is possible that they used the application more often than they would have normally, which is commonly known as the "Hawthorne Effect" [MWE]. Nonetheless, participants had use cases for the application, and the extent of this effect is considered limited. Another threat to the internal validity is that one participant was eliminated due to inactivity. This could have biased the study results, even though there were seven other participants. Nevertheless, it must be mentioned that one feedback would have had limited weight compared to the other seven. A further threat to internal validity could be that the author of this thesis was extensively involved in the creation of the codings for the thematic analysis. To reduce this threat to a minimum, all codings were discussed in a group of three researchers.

Chapter 7

Conclusion

Within the scope of this thesis, we have developed the ScreenCurator, a screenshot-based prototype in the field of information curation. Various information curation prototypes with related goals were analyzed to identify a set of valuable features. This prototype was specifically designed to support knowledge workers in tasks that include information foraging. The design goals were to offer an intuitive and semi-automated application that allows users to retrieve foraged and curated items effortlessly. Furthermore, a specific focus was set on capturing cross-application resources, for which screenshots are perfectly suitable.

The ScreenCurator was tested and evaluated in a user study by seven participants. The results indicate that the application improves the overall experience of taking and retrieving screenshots. On a high level, two different use cases could be identified: Long-term backups and short-term to-dos. On a lower level, almost every participant used the ScreenCurator differently and completed various tasks with it. Even though the study participants captured mainly browsers, most of them took screenshots of other applications as well. This underlines the importance of the ability to store assets from different sources together. To enable all the diverse use cases, the ScreenCurator must be highly flexible. To achieve this flexibility and cater to all user preferences, several enhancements can be implemented. Consequently, the feature set would need to be slightly extended. One of the most important enhancements would be the extension and improvement of existing automation features. Further tag suggestions for screenshots and automatic URL extraction in a web setting are also highly demanded features. Furthermore, additional structuring and joining options for screenshots are required. Another valuable addition to the feature set would be a collaborative interface, including collection-sharing options to enable team-based information reuse. Nevertheless, it should be ensured that the application remains, despite new features, simple and intuitive. Lastly, the functioning and quality of implemented features should always be ensured.

This work's primary contribution involves the design and development of a screenshot-based information curation application, as well as its evaluation for effectiveness. This evaluation included the analysis of the experience, the possible use cases, and the feature set, which is needed to provide a functional application. In further work, an enhanced prototype could be developed, which implements the possible improvements found by the conducted study. Additionally, a long-term user study analyzing the coexistence of long-term and short-term assets inside screenshot-based information curation applications would be of great value.

Appendices

Appendix A

Consent Form

University of Zurich^UZH

Human Aspects of Software Engineering

University of Zürich
Department of Informatics
Binzmühlestrasse 14
CH-8050 Zürich

Contact Person:
Prof. Dr. Thomas Fritz
Tel: +41 44 635 67 32
fritz@ifi.uzh.ch

Consent Form – “ScreenCurator” Study

Principal Investigator

Christoph Bachmann, Dept. of Informatics, University of Zurich, christoph.bachmann3@uzh.ch

Collaborators

Roy Rutishauser, Dept. of Informatics, University of Zurich, rutis@ifi.uzh.ch
Alexander Lill, Dept. of Informatics, University of Zurich, alexander.lill@uzh.ch
Prof. Dr. Thomas Fritz, Dept. of Informatics, University of Zurich, fritz@ifi.uzh.ch

Purpose

In recent years, the amount of information that modern knowledge workers are confronted with is constantly increasing. To efficiently use found information, they should be stored, maintained, and reused in appropriate cases. One of the easiest ways to capture a cross-application setting is to take a simple screenshot. Screenshots are also a reliant and low-effort way of capturing the current state of a desktop. The problem with this method is that it is time-intensive and laborious to organize screenshots if the user plans to reuse them. The aim of this study is to examine an approach we developed to support knowledge workers in their struggle to reuse previously found information and evaluate whether it can help to improve the way how they forge for knowledge.

Study Procedure

Overall, the study spans 10-15 workdays and consists of the following steps and phases:

Setup Session: After receiving the installation folder, the participant shall install the application on a computer. This session should take **no longer than 20 minutes** of your time, including a short introduction.

Trial Phase: The participants are expected to continuously use the ScreenCurator application during the mentioned span of 10-15 working days for daily information seeking related tasks.

Interview Session: After the testing phase participants are asked to participate in an interview to give feedback on the usage and experience with the application. This last part, including the interview or questionnaire will take **no longer than 45 minutes.**

Benefits and Risks

By participating in this study, you will have the chance to test a newly built application that shall help you to be more efficient in information seeking and reuse. This could help you to boost your work performance.
The main risk is the loss of time required to participate in the study. The total amount of time for participating in the study will be approximately 4 hours over the 10-15 workdays. We are mitigating this risk by allowing you to determine a suitable time for the study participation as well as the sessions with the researchers. Furthermore, you are free to withdraw from participation at any point during the study, without the need to provide a reason.

Data Collection

During the study, our approach collects data about your interaction with the application, but neither clear-text user content nor screenshot images leave your laptop and will not be reviewed. All data is stored locally and will never be uploaded to the internet. The entire application works offline.

ScreenCurator Interaction Data: ScreenCurator saves log-data about the interaction like when a screenshot is taken, a search is performed, or a filter is applied. At the end of the study, we will ask you to share an obfuscated version of the interaction data and screenshot metadata (no plain text data leaves the application). You will be able to review the data before sharing.

Data Storage and Confidentiality

All data will be treated confidentially and only reside on machines of the university researchers. Your data will be used and seen only by researchers directly involved with this project. Interviews may be audio recorded (if you consent) and will be transcribed by a university-approved professional transcription service (if you consent). After the transcription is created, the audio recording will be deleted. In case there are any identifying information in the transcripts or any other collected data, the researchers will manually remove them before starting the data analysis.

In any case, raw data may not be stored for longer than one year. All identifying information will be kept strictly separate at all times from any other collected data and will be stored in a different location at the University of Zurich. Only the involved researchers will have access to this information. Furthermore, it will not be associated with the data after it has been analyzed. After the study is finished, your identifying information will be permanently deleted. The anonymized, non-identifiable data produced from the study will be stored for five years, after which it will be permanently and irrevocably deleted.

Uses of the Study Data

The results of this study will potentially appear in both internal and external academic research presentations and publications, such as academic journals and conference proceedings.

Contact for Information about the Study

If you have any questions or desire further information with respect to the study, you may contact Christoph Bachmann (Christoph.bachmann3@uzh.ch), Alexander Lill (alexander.lill@uzh.ch), Roy Rutishauser (roy.rutishauser@uzh.ch) or Prof. Dr. Thomas Fritz (fritz@ifi.uzh.ch).

Consent for extended Data Uses

With your explicit consent, you can allow the researchers to transcribe the audio recording of the interview using a university-approved transcription service:

❑ **I allow the use of a transcription service to transcribe my interview.**

Consent for Study Participation

Your participation in this study is entirely voluntary. You are free to withdraw your participation at any point during the study, without giving any reason and without any negative consequence. Any information you contribute up to your withdrawal will be retained and used in this study, unless you request otherwise.

With your signature on this form you confirm the following statements:

- I understand the goals and procedures of the study and the applicable conditions.
- I had the opportunity to ask questions. I understood the answers and accept them.
- I am at least 18 years old.
- I had enough time to make the decision to participate and I agree to the participation.

In no way does this waive your legal rights or release the investigators or involved institutions from their legal or professional responsibilities.

____

Participant’s name

________________________________ ____________________________________

Location, Date Participant’s signature

Appendix B

Pre-/Post Questionnaire

Pre-Questionaire

This is the pre-study questionnaire for the ScreenCurator user-study.

1. Screenshot Usage

	Never	Rarely	Sometimes	Often	Always
How often do you take screenshots?	○	○	○	○	○
How often do you retrieve old screenshots?	○	○	○	○	○

2. Experience

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
I find it laborious to manage and organize my screenshots.	○	○	○	○	○
I can easily refind my screenshots when I need them.	○	○	○	○	○
I structure my screenshots to ease future retrieval.	○	○	○	○	○

3. For what purposes do you use Screenshots?

4. What do you think are the advantages of images as a medium to store knowledge?

5. What do you think are the disadvantages of images as a medium to store knowledge?

This content is neither created nor endorsed by Microsoft. The data you submit will be sent to the form owner.

Microsoft Forms

Appendix C

Interview Questions

Interview Questions

Demographics

- Ask for gender, age, profession, type of tasks/work the user has done while using the ScreenCurator.

Purpose

1. How was your general experience with the ScreenCurator?
2. How did you use the ScreenCurator?
    a. Could you scroll through your collection and tell me what you see on the screenshot and why you took it?
        i. Now that you told me why you took these screenshots could you tell me some other reasons why you took screenshots?
    b. Would you like to share, which applications were open most of the time when you took screenshots?
    c. Do you remember specific situations when working with the ScreenCurator was especially frustrating or useful?
3. How did you use the screenshots after taking them? E.g. as Todo items, information repository, …
4. Do you think you would use these screenshots in the future?
    a. Could you imagine using them as a repository of knowledge?
    b. How would you use them?
    c. Please name some situations when you would specifically search this repository of knowledge.
5. For how long do you think you would keep Screenshots in average -> Over days, weeks, months, unknown?

Features

1. What features did you particularly like or dislike?
    a. Like search, filter or freehand-drawing
2. Did you tag your screenshots?
    a. Why did you tag your screenshots and did you have a particular system behind the tags
    b. Would you prefer a folder structure?

i. Why?

3. Do you remember a situation where the freehand drawing tool was especially useful or frustrating?
4. Are there any essential features missing?
   a. Could you think of any other useful features that could be added?

Retrieval

1. Did you retrieve any screenshots? How did you mainly retrieve elements from the library? -> Hierarchical questions
   a. Did you use Filters (which?), Text search or scrolling?
   b. Did you use different retrieval techniques in different situations?
   c. If scrolling search was done: How, do you think, would you primarily retrieve screenshots when the library gets larger?
   d. Did you remember a screenshot retrieval that was especially difficult? -> If yes, how did you tackle the challenge?

Appendix D

Visible Windows Algorithm Explanation

First, the algorithm detects on which monitor a screenshot is taken, scales, and transforms the borders of the applications accordingly. Afterward, the window is reduced to a list of corners, which includes at the beginning the four corners indicated by the application border.

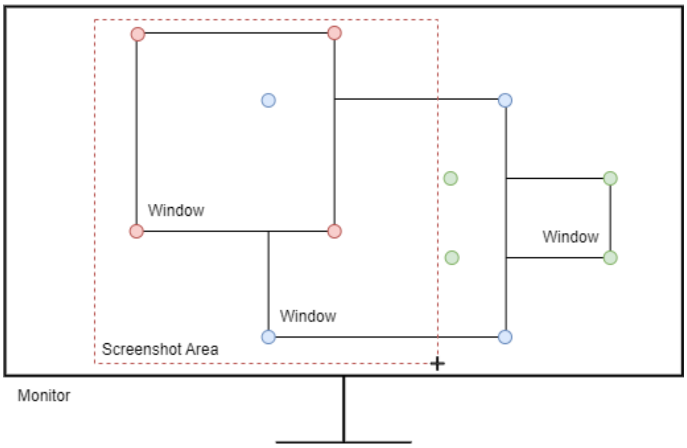


Figure D.1: Visible Windows Algorithm: Windows with corresponding corners

Then for every window, it is checked if one corner point is located in the screenshot area. If this is not the case, the window is discarded. Else, all the window corners outside the screenshot area are projected onto the screenshot area boundary.

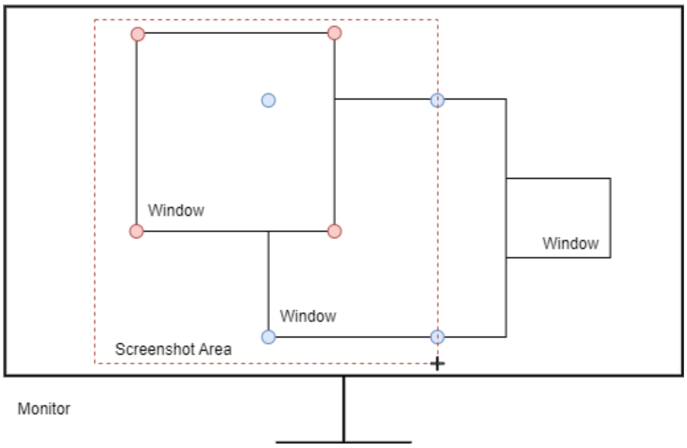


Figure D.2: Visible Windows Algorithm: Windows after corner projection

As a next step, the algorithm selects all corners inside the window area with a lower index (more in the front) and deletes them. Moreover, the algorithm creates new corner points for each intersection of the lower window with the upper window.

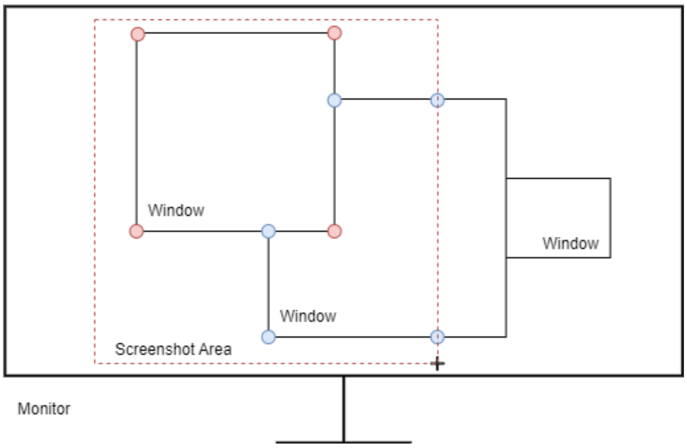


Figure D.3: Visible Windows Algorithm: Deletion of covered corners and creation of intersection corners

Finally, the algorithm adds corner points to the lower window for each upper window corner that is inside the lower window area.

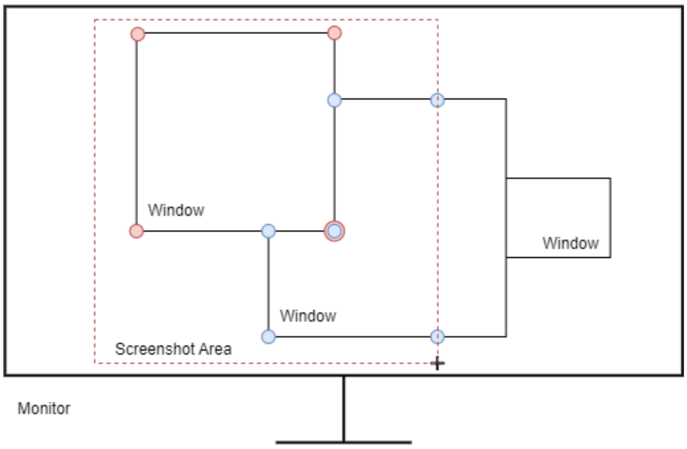


Figure D.4: Visible Windows Algorithm: Addition of upper corners into lower windows

If a window has less than four corners, it gets discarded from the list of visible windows.

Bibliography

[ABC98] David Abrams, Ron Baecker, and Mark Chignell. Information Archiving with Bookmarks: Personal Web Space Construction and Organization. *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 41–48, 4 1998.

[BC06] Virginia Braun and Victoria Clarke. Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2):77–101, 2006.

[BH11] P. Bryan Heidorn. The Emerging Role of Libraries in Data Curation and E-science. *Journal of Library Administration*, 51(7-8):662–672, 10 2011.

[BKAO08] Timothy F. Brady, Talia Konkle, George A. Alvarez, and Aude Oliva. Visual long-term memory has a massive storage capacity for object details. *Proceedings of the National Academy of Sciences of the United States of America*, 105(38):14325–14329, 9 2008.

[Bui] Build cross-platform desktop apps with JavaScript, HTML, and CSS | Electron. *https://www.electronjs.org/ accessed 16/02/2023.*

[Chu] Anthony Chuter. OneNote is my ultimate curation and organizational tool – OneNote in Education Part 1 of 3. *https://ict4kids.ca/2018/07/15/onenote-is-my-ultimate-curation-and-organizational-tool-onenote-in-education-part-1-of-3/ accessed 24/04/2023.*

[CJKB08] Andrea Civan, William Jones, Predrag Klasnja, and Harry Bruce. Better to organize personal information by folders or by tags?: The devil is in the details. *Proceedings of the American Society for Information Science and Technology*, 45(1):1–13, 1 2008.

[Cre] Create a new Gist. *https://gist.github.com/ accessed 02/08/2023.*

[Dal14] Stephen Dale. Content curation. *Business Information Review*, 31(4):199–205, 1 2014.

[DMS99] Wei Ding, Gary Marchionini, and Dagobert Soergel. Multimodal Surrogates for Video Browsing. In *Proceedings of the fourth ACM conference on Digital libraries*, pages 85–93, 1999.

[Fre15] Luanne Freund. Contextualizing the information-seeking behavior of software engineers. *Journal of the Association for Information Science and Technology*, 66(8):1594–1605, 8 2015.

[Git] GitHub - electron-react-boilerplate/electron-react-boilerplate: A Foundation for Scalable Cross-Platform Apps. *https://github.com/electron-react-boilerplate/electron-react-boilerplate accessed 04/07/2023.*

[HAS] HASEL-UZH/screen-curator. *https://github.com/HASEL-UZH/screen-curator accessed 02/08/2023.*

[HL20] Donghan Hu and Sang Won Lee. ScreenTrack: Using a Visual History of a Computer Screen to Retrieve Documents and Web Pages. 2020.

[HL22] Donghan Hu and Sang Won Lee. Scrapbook: Screenshot-Based Bookmarks for Effective Digital Resource Curation across Applications. pages 1–13, 10 2022.

[KACB03] John M Kamara, Chimay J Anumba, Patricia M Carrillo, and Nasreddine Bouchlaghem. Conceptual framework for live capture and reuse of project knowledge. *cs.auckland.ac.nz*, 284:47–55, 2003.

[KAM$^+$21] Rebecca Krosnick, Fraser Anderson, Justin Matejka, Steve Oney, Walter S. Lasecki, and Tovi Grossman. Think-aloud computing: Supporting rich and low-efort knowledge capture. *Conference on Human Factors in Computing Systems - Proceedings*, 5 2021.

[KCH$^+$22] Andrew Kuznetsov, Joseph Chee Chang, Nathan Hahn, Napol Rachatasumrit, Bradley Breneisen, Julina Coupland, and Aniket Kittur. Fuse: In-Situ Sensemaking Support in the Browser. *UIST 2022 - Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*, 10 2022.

[KLL$^+$17] Andruid Kerne, Nic Lupfer, Rhema Linder, Yin Qu, Alyssa Valdez, Ajit Jain, Kade Keith, Matthew Carrasco, Jorge Vanegas, and Andrew Billingsley. Strategies of Free-Form Web Curation: Processes of Creative Engagement with Prior Work. 2017.

[LF94] Mike Lamming and Mike Flynn. "Forget-me-not" - Intimate Computing in Support of Human Memory, 2 1994.

[LKM22] Michael Xieyang Liu, Aniket Kittur, and Brad A. Myers. Crystalline: Lowering the Cost for Developers to Collect and Organize Information for Decision Making. *Conference on Human Factors in Computing Systems - Proceedings*, 4 2022.

[MWE] Jim Mccambridge, John Witton, and Diana R Elbourne. Systematic review of the Hawthorne effect: New concepts are needed to study research participation effects.

[NK18] Klaus North and Gita Kumta. *Knowledge Management*. Springer Texts in Business and Economics. Springer International Publishing, Cham, 2018.

[Pasa] Pastebin.com - #1 paste tool since 2002! *https://pastebin.com/ accessed 02/08/2023.*

[Pasb] Pastebin.com - FAQ [Frequently Asked Questions]. *https://pastebin.com/faq accessed 02/08/2023.*

[Pen07] Maureen Pennock. Digital Curation: A Life-Cycle Approach to Managing and Preserving Usable Digital Information. *Library & Archives*, 1:1–3, 2007.

[Pir07] Peter Pirolli. *Information Foraging Theory: Adaptive Interaction with Information*. Oxford University Press New York, 2007.

[Poo17] Alex H. Poole. "A greatly unexplored area": Digital curation and innovation in digital humanities. *Journal of the Association for Information Science and Technology*, 68(7):1772–1781, 7 2017.

[ROH19] Søren Rasmussen, Jeanette Falk Olesen, and Kim Halskov. Co-notate: Exploring real-time annotations to capture situational design knowledge. *DIS 2019 - Proceedings of the 2019 ACM Designing Interactive Systems Conference*, pages 161–172, 6 2019.

[RTBH15] Adam Rule, Aurélien Tabard, Karen Boyd, and Jim Hollan. Restoring the Context of Interrupted Work with Desktop Thumbnails. *Cognitive Science Society*, pages 1–6, 2015.

[SFM20] Chris Satterfield, Thomas Fritz, and Gail C Murphy. Identifying and Describing Information Seeking Tasks. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, volume 12, pages 797–808. Virtual Event, Australia. ACM, 2020.

[SQL] SQLite Home Page. *https://www.sqlite.org/index.html accessed 07/05/2023*.

[Tes] tesseract-ocr/tesseract: Tesseract Open Source OCR Engine (main repository). *https://github.com/tesseract-ocr/tesseract accessed 07/05/2023*.

[Tra23] Duc-Hieu Tran. Screenshots Enriched with Metadata to Enable Bookmarking and Semantic Retrieval. 2023.

[WGK10] MO Ward, G Grinstein, and D Keim. *Interactive data visualization: foundations, techniques, and applications*. CRC press, 2010.

[Wha] What is Fuse.js? | Fuse.js. *https://fusejs.io/ accessed 05/07/2023*.

[Whi11] Steve Whittaker. Personal Information Management: From Information Consumption to Curation. *Annual review of information science and technology*, 45:1, 2011.

[WPCWG15] Weiliang Wang, Germán Poo-Caamaño, Evan Wilde, and Daniel M. German. What is the GIST? Understanding the use of public gists on GitHub. *IEEE International Working Conference on Mining Software Repositories*, 2015-August:314–323, 8 2015.