SIMPLSHOT: An easy way to share a snippet of a webpage on the social web

Sukriti Sharma
North Carolina State University
Raleigh,NC
ssharm18@nscu.edu

Arjun Thimmareddy
North Carolina State University
Raleigh,NC
athimma@ncsu.edu

Ankit Murarka
North Carolina State University
Raleigh,NC
amurark@nscu.edu

Sushma Ravichandran
North Carolina State University
Raleigh,NC
sravich@nscu.edu

ABSTRACT

With the advent of social media, people spend a lot of time browsing the web and tend to share any interesting facts with their near and dear ones. However the ease with which you can share a snippet of the screen has not evolved as much. There are quite a few insipid approaches taken to solve the issue of sharing a snippet of a web page in the form of images on any social media. Our traditional way of taking a screenshot constitutes multiple steps and saving these screen shots on our local machine takes away unnecessary memory of our machines. We are looking at an innovative approach of solving this problem by reducing clutter and cumbersome task of taking screen-shots, cutting and sharing. We have designed three solutions each with its own set of features to ease the existing methods of sharing screen shots.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*observations, convenient measures*; H.4 [**Information Systems Applications**]: Human factor

General Terms

Observation, Case Study, Metrics

Keywords

SIMPLSHOT, Sharing made easy, text tagging

1. INTRODUCTION

The problem with sharing the a specific snippet of a screenshots with any of the social media across platforms has its way of causing inconvenience by involving multiple steps, taking up unnecessary space on the device and locating the saved image on the machine for uploading. The usual layman's way is to use 'PrtScr' or a desktop application to grab images and finally save them on our system. Following this, is the task of cutting, uploading the image using the upload button. Somewhat loosely, these steps of capture, save/edit and share are holy trinity of sharing the screen shots across platforms. We have performed an extensive study to identify the drawbacks of this usual approach. We conducted a survey and with considerate amount of observation to understand and identify how people get around this task. Our video recordings, interviews and surveys had targeted the audience with questions which subtly reveal the issues faced by them. The metrics indicate that a wide section of the people consider asking for a simple new way overcoming the drawbacks. Although a sparse population is aware of screen shot applications, not many use it due to the fact that it requires saving those files onto the local system.

After going through the user responses from the survey we designed three different solutions (two desktop applications and one Chrome plug in) to overcome the generic problems faced by the users who use the existing screen capturing tools. The common features of all three solutions is that the user can sign up, log in, private image capturing, public image capturing and providing user reviews. Apart from these two solutions, handle public and private searches for each user. The desktop applications extract textual content from each image and store it along with the image under the respective user credentials. The Chrome plugin, on the other hand uses the technique of manually tagging each image just before uploading a snippet. It also has the provision of editing the tags of each image. The enhanced desktop application and the Chrome plugin have a search tab where each user can search his/her private database or the public database for images (through textual contents or tags respectively).

Once we completed the design of each of the three solutions, we again targeted users who were willing to review our applications. We provided the users with a feedback form which asked them to rate the app on a scale of 1 to 5 and provide a few lines of comments as feedback. The willing users provided us with the feedback which helped us analyze the pros and cons of each solution. It also helped us analyze the general experience(what all problems they faced and what all features they found useful) of the user while using the 3 solutions and completing the tasks which was assigned to them as part of the survey. Through the comments and rating, we analyzed that Solution 2(Chrome Extension) was the most popular amongst the users.

2. ARCHITECTURE AND DESIGN

The different components of the application are being explained in brief here. This gives us an idea on the number birds eye view of the architecture in Fig 1.

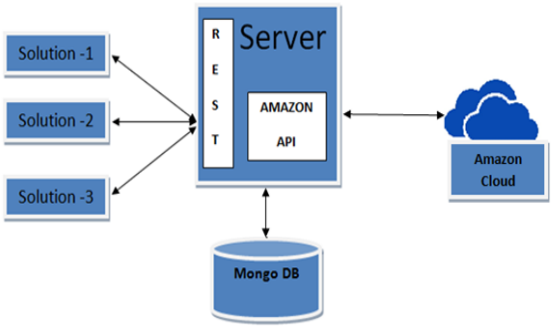

Figure 1: Architecture

2.1 Server

The server was built with the purpose of facilitating the upload and retrieval of user data from a centralized location. It servers major functionalities in provisioning the client application with the power to search, upload files and tag images. All of these functionalities have been explained in the above in the section for each of the solutions.
The technical aspects of the server were decided and the technologies used were based on the ease of maintaining and robust architecture design. The REST interfaces were designed to allow the three different approaches of the client application to interact with the server using REST get/post requests. The server maintains user data on the mongodb and performs authentication to maintain integrity of the user data. The user uploads the screenshot taken via the REST service and the server uses Amazon API to upload the images into the cloud. The next set of requests to fetch images related to a tag is achieved is limited to the server and no requests are sent to the cloud as we retain user and image tag data on the mongodb.
Auto tagging is a feature which is enabled for one of the solutions, which allows the user to only upload the image. The google tesseract API does OCR and tags the image with the text extracted. This allows us compare three different solutions in identifying which is the best way to allow users to pick screenshots.

2.2 Client Solution-1(Desktop application with 'History')

In the first solution, we have provided a simple Desktop app to the users using which they can take snippets, which will be uploaded to our server. Once the image is uploaded, the users have an option of sharing it using a shareable link of the image provided to them. The user can view all the snippets previously taken by him and can obtain a shareable link for each image. This solution serves the purpose of taking snapshots and being able to share it easily, without first saving the images on local machine.
The solution is implemented using Pyside. While taking screenshot , a transparent window appears on which the user can drag his mouse to select the coordinates. The selected portion is uploaded to the server , with other user details. The user has options to share the image and view his history of snippets taken.

2.3 Client Solution-2(Chrome Plug-in)

The second solution was built keeping in mind application portability, ease of usage and customized tagging abilities. To make this possible, we developed a Google Chrome browser extension. For this, the technologies used were HTML5, CSS3, JavaScript, JavaScript APIs for Google Chrome Extension, Pure-Min, a CSS styling library and AngularJS, a JavaScript structural framework for dynamic web apps.
The application, enables the user to Sign Up and create their personal accounts on the cloud. They can then capture snippets of web pages inside Chrome and tag them with custom labels. The users can capture private snippets or ones they want to make available for cross search. These snippets can then be searched using tags.

2.4 Client Solution-3(Desktop application with 'Search')

Solution 3 was designed to highlight a feature we used to store images or screen clippings. As in the case of the other solutions, a user has to sign up on the app before logging in. Once inside the app, he can choose to take clips of his screen. He can can choose to take clips that are stored privately under his email address or he could also take snippets of page that will be stored publicly with other user's publicly stored clips.
When the images are uploaded, we use a package called tesseract to extract textual details from them and store it in the database under each image. Using this, a user can search for a clip he/she took a while ago through the textual content of the image. The user could again choose to search his own private database or he/she could 'cross search' i.e. search the database that is publicly shared by all users using the application. We propose that this would be useful for users who generally use a screenshot application to take images of text.

3. SALIENT FEATURES OF THE CLIENT SOLUTIONS

3.1 Client Solution 1(Desktop application with 'History')

As shown in the figure 1, once a user clicks on the Share option the last image uploaded by the user is available as a thumbnail along with a link to the image URL.
Using solution 1 a user can share images from his/her past image uploads. Each image is displayed as a thumbnail as shown in Figure 2.

The following are the salient features of Client Solution-1(Desktop Application with History):

- User Log In and Sign Up.
- Take private Image Snap
- Take public Image Snap
- Automatic storage of images after extracting each textual contents

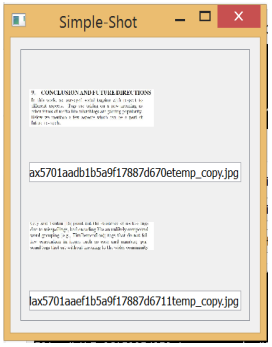

Figure 2: Solution 1- Search History

- Share previous image taken (with URL)
- Show History of Images of a user (with URL)
- User Review option

3.2 Client Solution 2(Chrome Plug-in)

Using solution 2 a user can add/delete custom tags to images. This is depicted clearly in Figure 3

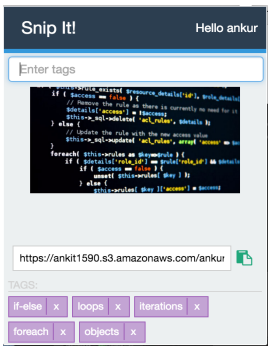

Figure 3: Solution 2- Image with tags

The user can then perform private search or a crosssearch based on the tags as shown in figure 4

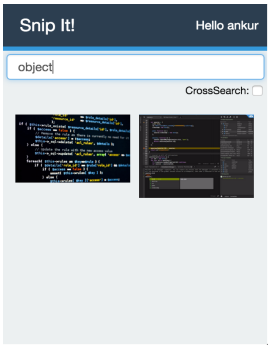

Figure 4: Solution 2- Search images using tags

The following are the salient features of Client Solution-2(Chrome Plug-in):

- User Log In and Sign Up
- Take Private Image Snap
- Take public Image Snap
- Manual tagging of images with user-specific tags
- Search history of privately stored images on through manual tags associated with each image
- Search history of publicly stored images on through manual tags associated with each image
- Retrieve private or public image URLs, copy to clipboard option and download image option
- Can manually delete or add tags to each image
- User Review option

3.3 Client Solution 3(Desktop application with 'Search')

Figure 5 is an image of the Share option available in Solution 3 where the user can have access of the URL to the most recently taken snip of the screen.

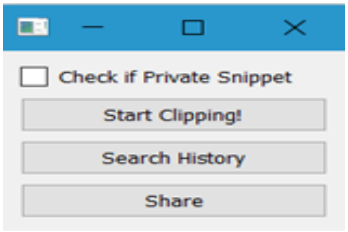

Figure 5: Solution 3- Share Image

The most prominent feature of Solution 3 is its search tab. A user can search through both his own private images as well as the images that are publicly stored under Cross Search. This is shown in figure 6

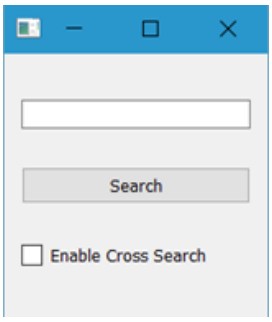

Figure 6: Solution 3- Search Tab

The following are the salient features of Client Solution-3(Desktop application with 'Search'):

- User Log In and Sign Up
- Take private Image Snap

- Take public Image Snap
- Automatic storage of images after extracting each textual contents
- Retrieve private Images by Search through textual content of each image so as to share with URL
- Retrieve public Images by Cross Search so as to share with URL
- User Review Option

4. EVALUATION

4.1 Methodology

To evaluate our solutions, we designed two tasks for the users that would help us get an overall rating for each solution. The tasks were designed to make the users explore the features in each solution. We aimed to get an understanding of which features would attract the users, which would help us rank the solutions relatively.

Task 1

The purpose of this task was to evaluate the effectiveness of the manual tagging mechanism in the Chrome Solution (solution 2) against the automatic tagging mechanism in the Desktop solution (solution 3). To make the users see he difference between the two approaches, we designed a tutorial for Lisp in our database. We took snippets of webpages with basic code snippets in Lisp. The snippets that were uploaded through Chrome plugin, had manual tags associated with them. The images uploaded through the Desktop solution, were automatically tagged by extracting the text from the images. All the images were made publically available. The users were given the challenge of writing a simple function in LISP to print the squares of the every even number below 10, by making use of our tutorial as the only source. The users would have to make use of the search facility, to retrieve the snippets for each feature (eg: for loop). While using the chrome plugin, the users search query will be matched with the manually added tags. While using the Desktop solution, the images are tagged automatically by our code, and the search query will be matched against those tags. The user will then evaluate our solutions based on their experience. Apart from the explicit feedback, we have added telemetry to collect data on the number of search queries the user makes, before he gets a hit. This will help us evaluate which of the tagging mechanism would yield better search results, based on how the users query for information. It would also show the user how his manual tags can be useful to someone else, when he publicly shares an image.

Task 2

The purpose of this task was to evaluate the basic snip and share feature of the Desktop solution (solution 1) versus the Chrome Plug-in (solution 3) The users were asked to take any snippet and share it over social media. By performing this task, the users would explore how to take snippets using both the solutions. The user can also explore the tagging mechanism in the Chrome solution. The users will use the links provided by the apps to share the images on social media. This task would give us a picture of whether the users find the Desktop solution more usable or the Chrome Plug-in. While performing both the tasks, the users would have explored the different features of each solution. Now the user is asked to give a relative ranking for each solution. The users will rank the solutions based on what features appealed to them the most. They also have an option to fill out their comments which would help us understand what majority of the users liked or disliked in each solution.

5. EVALUATION PLAN

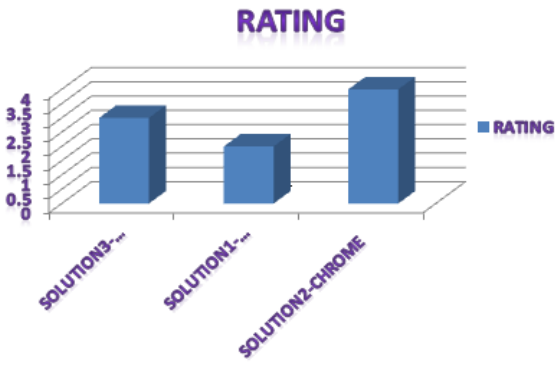

Figure 7: Application overall ratings

5.1 From User Reviews of Solution-1

The statistics show us that the average rating for solution 1 was only 2 . To understand why, we analysed the comments given by the users . While a few users claimed that the app had the necessary features they wanted, most of the users found it difficult to scroll and search for their images in the history . They claimed that they would prefer having a search option that will assist them in this task. Also, some users claimed that they would not prefer to use a Desktop app if a Chrome Plugin was available to perform the same functionality.

5.2 From User Reviews of Solution-2

After analyzing the feedback given by the users, we concluded that the average rating of solution 2 was 3. The users mostly felt satisfied after using the Chrome Extension and the reason cited by majority of them was the ability to add and delete custom tags to screen snippets. The number of search requests made by users to complete the task given to them were lesser than the number of search requests in the other solution. This proved that the users were more comfortable searching with custom tags rather than with text extracted out of the images. Another reason why people liked the Chrome Extension was that there was no need to download and install an application on their Drives.

5.3 From User Reviews of Solution-3

We found that the average rating that users gave Solution 3 was 4. This shows that although the application performed better than Solution 1 the users were not satisfied entirely by the tool. The features that were already present performed to their fullest extent but many users said that there still was some room for improvement. We also note that the number of search requests sent by each user during

the User task 2 was much more than the number of search requests in Solution 2. We analysed the responses given by the users and found that the surge in the number of search requests was because they had no control over the text associated with each image unlike Solution 2 where manual tagging was allowed. Hence, we can conclusively state that although the features in solution 3 was more enhanced than Solution 1, it did not cater to the needs of the users as much as Solution 2 did.The Fig.7 shows ratings overall.

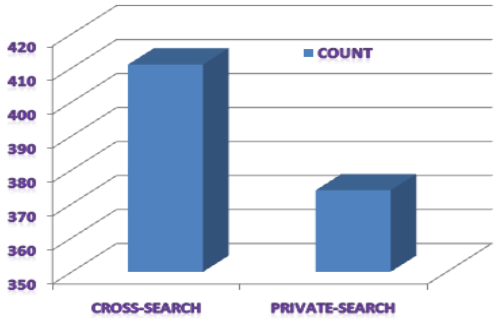

Figure 8: Categorized Search Count

The Fig.8 shows the count of two search facilities provided by our tools which were used by users to find images previously uploaded by them. Both Solution 2 and solution 3 had the provision of using private image search as well as cross search (on public images). Although our tasks specifically asked users to perform search on public images in the database, the users themselves explored both the private and public the search feature for both the tools. This shows us that the users find the search feature useful and would like to use it.

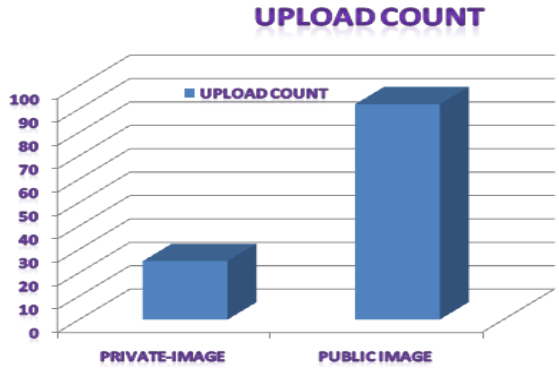

Figure 9: Image Upload count

By looking at the Fig.9, we observe that while users are more comfortable using the app to take public images that will be shared , they are also using the app to take private images that will not be visible to other users. This shows that distinction between private and public images was a useful feature for the users.

From the Fig 10 we observe that while performing the user task, the users had to search more on the Desktop solution

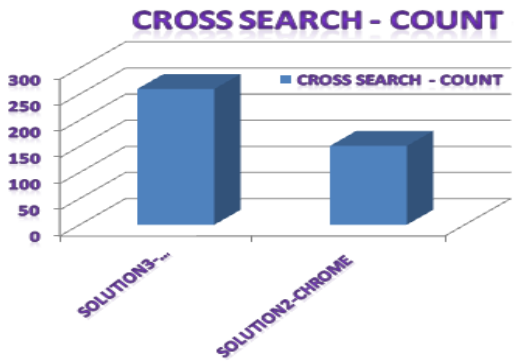

Figure 10: Cross Search Count

than the Chrome Plug-in, to find what they were searching for. This shows that the user queries match the manual tags given by other users more often than automatic tagging by extracting text from images.

6. CONCLUSION

In conclusion, we find that Solution 2 (Chrome Extension) proved to be the most popular one. We analyzed the comments entered by the users in the feedback and the most common reason was the ability to add custom tags in the Chrome Extension.

The driving factor behind the solution development was non-availability of a solid solution to allow users take and share snapshots. We had noticed the absence of a tool which surpasses the OS constraints and provide a uniform platform through which users could take, share and search the snapshots taken. We clearly noticed via our tests the inclination of users towards a browser based solution as they would not want the hurdle of installation of additional supporting application to be installed on their machines for the Desktop solutions though it provides quite an impressive list of features. Another key thing we noticed was the ability of the users to tag the images had a much higher USP than the automatic text extraction done via the Tessaract API. The flexibility for the user to add and delete the tags were something the user really looks for. They wasnt the ability to manually tag the images allowing them to recover them as and when they feel like via the search queries which they would have tagged with. Hence the analysis and tests concluded that the chrome solution was better than the rest of the solutions.

This was proved by both User task 1 and User task 2. User task 1 was designed to test the search features of Solution 2 and 3 whereas User task 2 was designed to test the image upload features of Solution 1 and 2. When it comes to uploading images, solution 3 had the option of manually tagging each image before uploading them. Texts from the images uploaded from solution 1 were automatically extracted and stored along with the images. After going through the user comments we found that the Solution 3 was far more user intuitive than Solution 1.

When it comes to search images from user history, we find that solution 3 was again more effective because the users tagged images according to their convenience while uploading. This gave them control over what was stored in the

database along with each image thereby making Solution 2 the most preferred one in all aspects.
Apart from this, the fact that Solution 2 pertains to the browser as an extension and does not have to be downloaded/installed as a tool, also goes in favor of the Chrome Extension hence making it the most popular solution out of the three.

7. REFERNCES

[1] http://osxdaily.com/2010/05/13/print-screen-mac.

[2] http://winaero.com/blog/how-to-take-a-screenshot-in-windows-8-1-three-ways-without-using-third-party-tools/.

[3] https://addons.mozilla.org/en-US/firefox/addon/imagegrabber