Undercover Recovery
Automated Social Media Open Source Intelligence
CSCE A470 Final Report Document
Tyler Lukacz and Logan Chamberlain
December 2022

1. Abstract

This project is a Win32 API based application, designed to automate the process of scraping web content. It consists of three parts, the first of which is the GUI, which allows a user to select and configure different options for the application to run. The GUI also calls and runs backend scripts and displays real time status updates. The second part of the project consists of python scripts to scrape the web content. There are four python scripts, one to authenticate user logon credentials and collect cookies, and then a tailored python script for each target website scraped for content. Lastly, the file structure for the project is configurable via the GUI to support either a local directory or scrape results can be routed to an external harddrive. The scraped content includes the links to a specific post, a link to the post or Tweet author, captions, timestamps when available, and single-image media.

2. Introduction

We built our tool for FBI- Anchorage VC1, with whom Logan is an intern, to assist with inefficiencies and potential intelligence gaps. Currently, social media analysis consumes a significant percentage of analysts' time conducting manual reviews of popular sites looking for suspicious posts, or posts that contain specific keywords or were authored by a known actor. These manual reviews are not only inefficient, but the limited capacity of a manual review leaves many posts unreviewed. To address this, we built an application that automatically searches Facebook, Instagram, and Twitter for suspicious activity and saves evidentiary copies of suspicious posts, saving analysts' time and allowing the Client to more holistically leverage social media to further the mission of protecting Alaska.

3. Project Overview

To implement this goal, we built a desktop application that periodically searches the most popular social media sites (Facebook, Instagram, and Twitter) looking for keywords and automatically saves copies of suspicious posts. Additionally, the application supports flagging accounts so that every new post made will be saved regardless of whether it contains keywords. The client is able to configure the keyword lists to tailor the product to specific investigations or institutional priorities, and can use the flagged users functionality to keep an eye on known actors that are of particular interest. To address the common tactic of suspicious posts being taken down quickly, the web scraping functionality downloads copies of the content onto the local drive. Recognizing this feature may impose substantial memory problems, the product supports the use of custom output directories to include external harddrives.

3.1 Security and Legal Considerations

As the project is being developed for use by a Government entity, security will be a priority. As the program will take and save social media login information, to include passwords, user configuration information is encrypted with a simple cipher prior to storage. We have also considered that certain API usage, such as the official Twitter API Tweepy, is specifically forbidden surveillance purposes. As such, while we recognize that use of APIs would allow for increased functionality, we are limited to emulating user interactions with the websites and will be conducting our scraping through web requests. We also note that all information retrieved is publicly available and within bounds for legal collection for this purpose. Additionally, we have included a rigid End User License Agreement to restrict the use cases of our tool that is discussed in a later section.

3.2 End User License Agreement

To manage the ethical complexity of building what is at its core a surveillance tool, we have included a rigid End User License Agreement (EULA) that dictates the terms of the appropriate use cases and installations. We, the authors, retain all rights and all modifications, additional installations, or integrations with other programs must be approved by us. We have imposed these limitations to ensure that our tool is used as intended and we may revoke rights to an installation if used improperly. See the End User License Agreement file included as part of installation for full terms.

4. Use Cases

There are two intended use cases for our tool that feature different configurations and goals.

In the Reactive use case, should a major event occur in Alaska that the user is interested in collecting as much social media information as possible on, we recommend configuring the tool to look for keywords that are directly related to the event in question and set it to scan every 3 hours. This method is an implementation of the deep but narrow ideology as the specific keywords usage limits the content to only content most related to the event, but the shortened scan frequency collects more information.

The Proactive, more general, use case, configures keywords to look for words of interest, the flagged users lists to look for known actors, and the scan frequency to 1 day to 1 week (24 - 168 hours). In this use case, the tool will ambiently monitor the target sites and download information of interest. The tool will automatically reauthenticate upon cookie expiration, meaning the user only has to login on the initial launch of the tool. This is an implementation of the wide but shallow ideology as it collects information less frequently, but hits a broader range of actors and keywords.

5. Requirements

The requirements established by the client were broad and non-technical, and thus an English expression of the requirements was best suited to capture the general idea of the product the client is

looking for. The core product focuses on automatic data collection, with future iterations of this project built on performing analyses of said data. The minimum viable product (MVP) was 1) a graphical interface that 2) collects login information, wordlists, region information, and scan frequency that then 3) scrapes social media sites using that login information and looking for posts containing keywords or posted by flagged users and finally 4) collects evidentiary copies of posts that meet those criteria. This MVP was developed for Instagram functionality by Mid October 2022, with Facebook and Twitter functionality as well as GUI beautification efforts filling the latter part of the semester.

5.1 Functional Specifications

- An Authenticator script takes usernames and passwords from users for each site and saves this information on a successful login.
    a. User credentials are encrypted using a cipher so as to avoid saving usernames and passwords locally in clear text.
    b. This script saves the acquired cookies for use by later scraper programs.
- The GUI allows the client to specify a time interval between scans that resets after each scan until the user-defined scan limit is reached or in perpetuity.
- The GUI configuration panel allows users to specify a custom output directory or use the default directory.
- The GUI configuration panel directs the user to a keywords file which contains instructions on proper use.
- Advanced search functionality is supported for Instagram scraping to include imposing a date limit and requiring a post contains multiple keywords to be considered a match.
- The GUI allows the selection of one of five predefined regions, namely Anchorage, Juneau, Fairbanks, Bethel, and all of Alaska. These region selections reference CSV files that contain lists of urls that filter the data by region. Location tags must be specifically defined with Instagram resulting in a much larger number of locations, whereas Facebook and Twitter sort posts into cities. The Program supports a total of 644 Instagram location tags, 76 cities recognized by Facebook, and 45 cities recognized by Twitter.
- Program scrapes web information from Facebook, Instagram, and Twitter through an automated chromedriver web browser powered through Selenium.
- The Program aggregates all the relevant information from each post of interest depending on the site the content was sourced from, generally including post author, caption, timestamp, and media, and writes reports in an html format to a specified output directory.
- Program displays real time updates from each respective site to the GUI including how many locations have been scanned, how many remain, how many posts of interest have been found, and how many total posts have been searched. For Twitter and Facebook, the program will scroll to reach the end of the results for each specific query, and displays how many scrolls were executed. At the end of each run, a summary of each scan is printed that includes the number of posts searched and number of posts of interest found and saved.
- The GUI includes an “Export Full Scan Results” button for each supported site that opens the file locations of the completed scan html documents.

- ➢ The GUI includes a timer that counts down from the user-established scan frequency. This serves both to inform the user of the remaining downtime and to let the user know the program is still running.
- ➢ The GUI has a modular design: each site has its own login button, flagged users list button, scan summary section that prints scraper results in real time, and export buttons.

6. Design

6.1 Graphical Interface

Figure 1 below shows the User Interface (UI) while running a scan. The timer displays how long until the next scan, the scan count displays how many scans have been conducted, each site has its own flagged users list accessible via the Flagged Users buttons, their own login button which displays the text Running while a scan is underway, their own button that references the directory where the scan reports are being written to called Export Full Scan Results, and finally a Launch button. Upon initially opening the program, the Launch button and all site Login buttons will be red, indicating the sites have not been logged into and the program may not be Launched. After logging into a site, the Login button turns green and displays the text ‘Set’ indicating a successful login. After at least one site has been logged into and the configuration has been set, the Launch button turns green indicating the program is set.

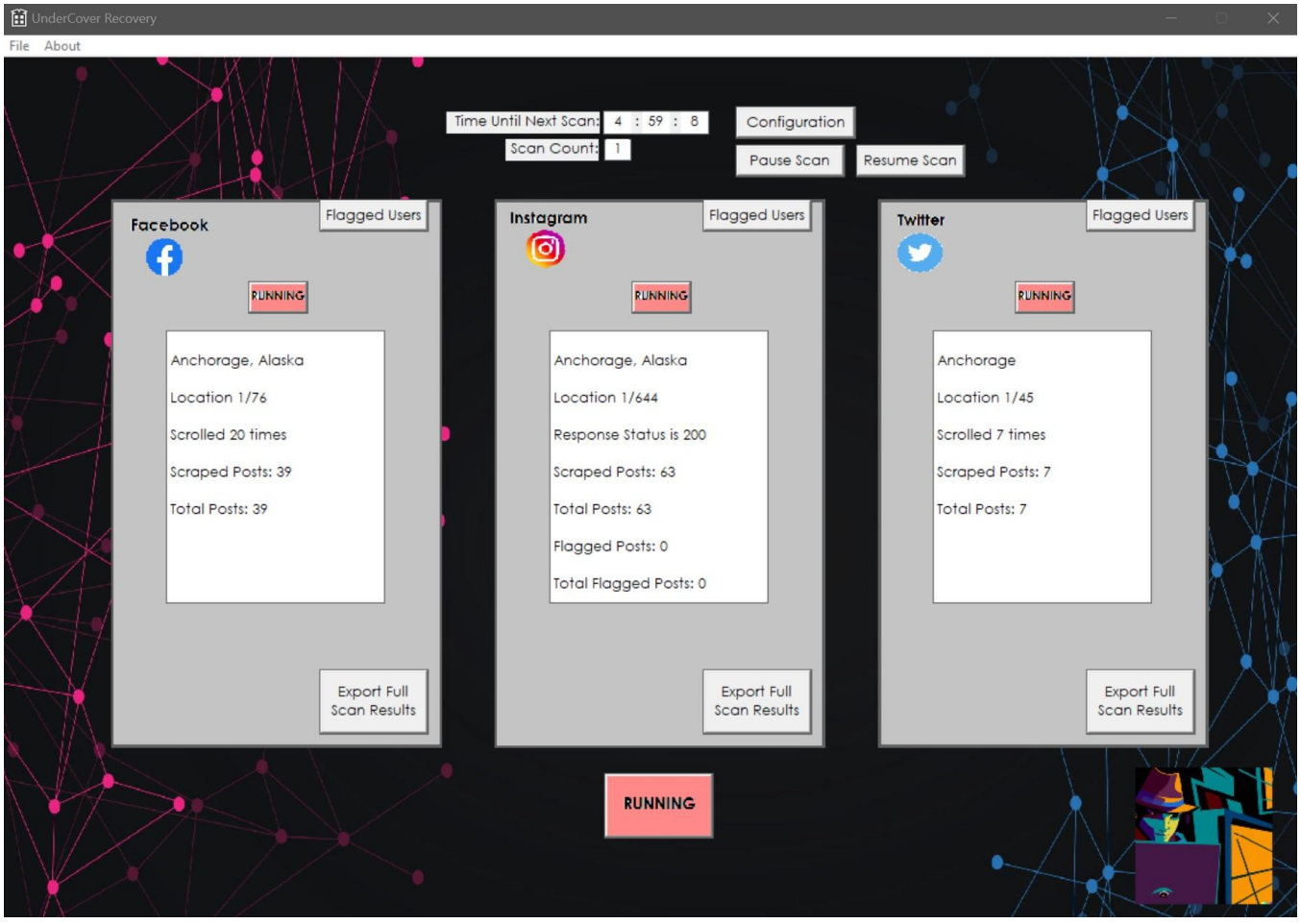


Figure 1, UI Screenshot while conducting a scan

Configuration options are set in the configuration panel, allowing the user to set a scan frequency, custom output directory, scan limit, and configure keywords.

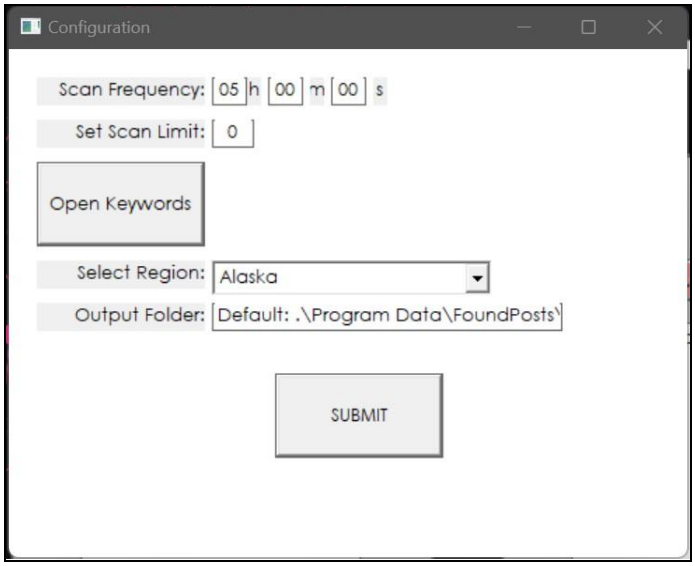
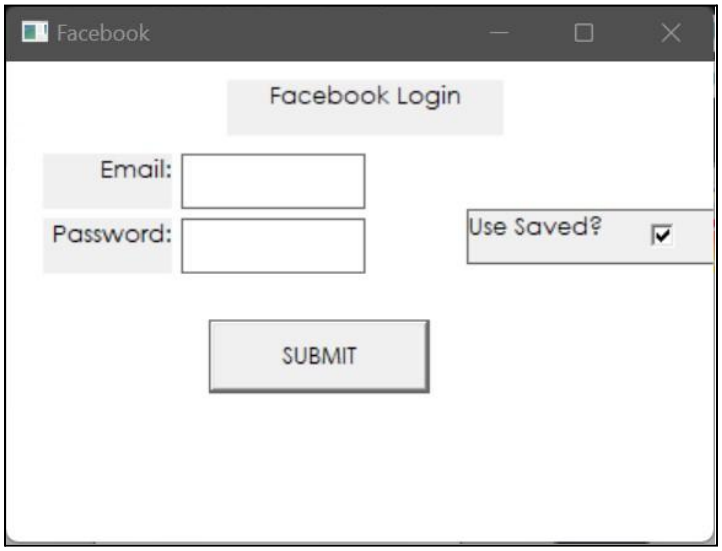


Figure 2, UI Screenshot of configuration panel (Right)



The Login button for each site opens a login panel where users are prompted for a username, or email in the case of Facebook, and a password. After a successful login, the credentials are encrypted using a simple cipher and stored, allowing the user to simply select the 'Use Saved' checkbox to expedite future logins.

Figure 3, UI Screenshot of Login panel for Facebook (Left)

6.2 System Architecture and Device Requirements

The client uses Windows machines for current social media reviews, so to be consistent with their current hardware, we built a Win32 Application. As such, the executable file must be run on a Windows machine running at least Windows 2000. As evidentiary copies are saved, should the keywords list include very common words or other circumstances leading to many saved posts, there is the potential for large local storage requirements. This will depend on the usage and can be mitigated by allowing users to set a custom output directory, to include external storage. Additionally, many of the web scraping functions and file handling will be conducted through Python scripts, so the client device must have

Python installed. Project-specific libraries are checked and installed if needed internally in the python scripts, but at a minimum a recent version of Python must be installed prior to running the application. The program also uses a chrome web driver of version 108.0.5359.99 and therefore requires that the client device have the same version of Chrome installed.

6.3 Data Structures and Algorithms

The primary and most common data structure used are Python lists. We use lists to store keywords, usernames, urls, and flagged posts. We also use Python dictionaries to resolve information passed as command line arguments to the scripting scripts into more verbose information. Our core algorithms are scripts for attempting logins and retrieving session cookies, web scraping, filtering data, searching return data, and writing outputs. While backend processing is being handled by python scripts, the GUI reads from log files to provide real time updates of the progress.

6.4 System Architecture and Components

The system is built into the modules shown in Figure 2. Users enter the system by launching the GUI which collects information that is passed to backend scripts for authentication and web scraping. The results are processed and real time scrape updates as well as a final summary are posted to the GUI and full results written into a local file.

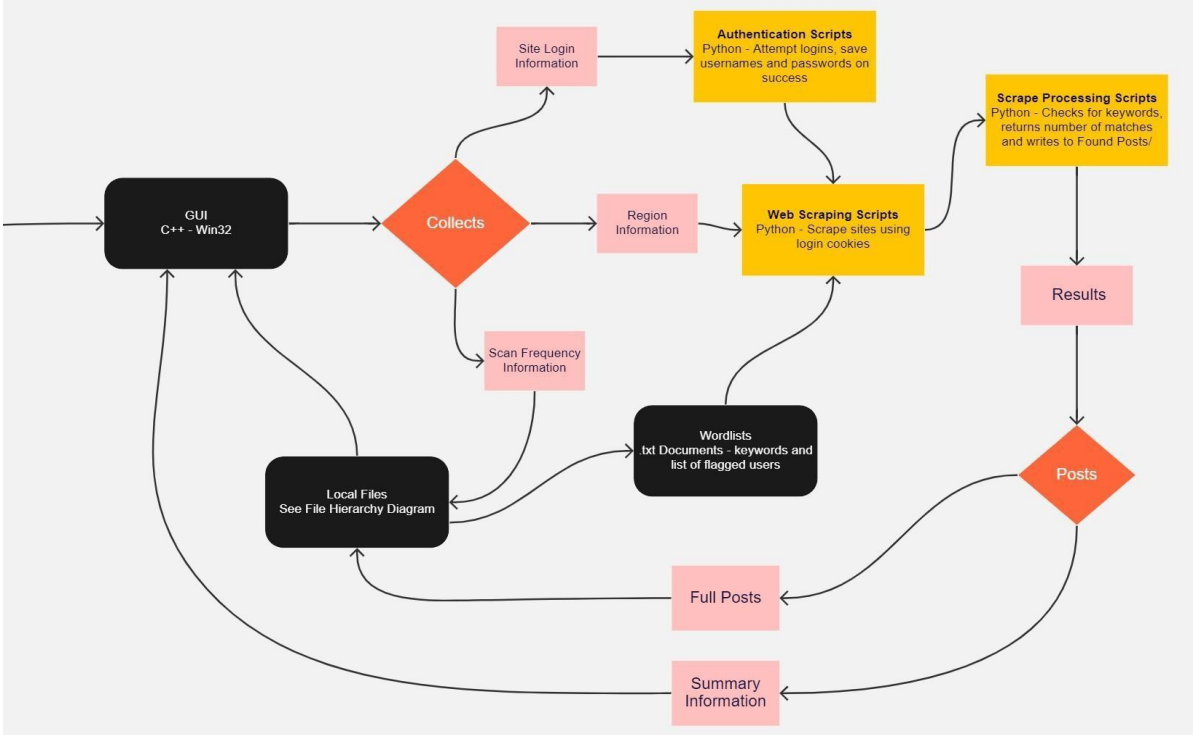


Figure 4, System Components and Data Flow Diagram

6.5 Regions

To limit the volume of posts searched, users are able to constrict their radius to one of several pre-set location options. These options, as our client is primarily concerned with posts originating in Alaska, will include Alaska as a whole or specific cities such as Anchorage, Fairbanks, Juneau, and Bethel. These city lists are only a subset of the locations contained in the Alaska (all) list. Instagram stores post information by specific location ID, for example to see posts tagged to 'Anchorage, AK', a request would be sent to: https://www.instagram.com/explore/locations/760995699/. Whereas Facebook and Twitter group posts by city, generally within a 20 mile radius. As such, a far more granular list of Instagram locations was curated, 644 in all, while fewer cities, 76 for Facebook and 45 for Twitter, were required to cover the same geographic area. These urls are included as part of installation (see Figure 7 File Hierarchy for specific locations). Once a region is selected from the drop down menu in the main GUI, the selected list is passed to the web scraping functions and each is searched in turn according to the respective site protocol.

6.6 Web Scraping

Web scraping is performed in Python using the Selenium and BeautifulSoup libraries. These scripts use a cookie obtained from the authentication scripts and the region list selected in the configuration panel. Facebook and Twitter are structured similarly and follow a similar protocol, however Instagram differs and has a separate protocol detailed below.

6.7 Instagram Scraping Protocol

Instagram allows users to see all posts tagged to a specific location. As such, at a high level the protocol for scraping Instagram is to 1) Load the location IDs from the region CSV into a list, then 2) iterate over that list and scrape information from each location. Finally, this mass of posts is filtered looking for posts with captions that contain a keyword followed by a punctuation mark or space (this is designed to reduce false positives), or posts that were authored by accounts from the Flagged Users list. Posts matching either of these criteria are parsed for the most relevant data, formatted, and written into html reports. The Instagram scraper collects the date/time a post was made, the latitude and longitude it was posted at, the username of the post author, the Full Name of the post author, the caption of the post, and downloads single image media. Additionally, it creates links to the author's profile and a link to the specific flagged post. A full scan of all collected Alaskan location tags takes an average of 2+ hours and processes an average of 20,000 posts.

6.8 Facebook and Twitter Scraping Protocol

Facebook and Twitter both do not directly allow users to see all posts tagged to a specific location, but this information can be accessed through crafted urls. The process for these sites is to first load the keywords and flagged users into prefabricated urls that contain filters that specify the city to be scanned, then iterate through and send requests to the sites looking for posts under these urls. If a url has

no results it means that either the location in question had no relevant results for that keyword, otherwise if a post rendered under that url meets the flagging criteria. As an example, to send a request to Facebook looking for the word “keyword”, in Anchorage, sorted by most recent, the program would get the following url:

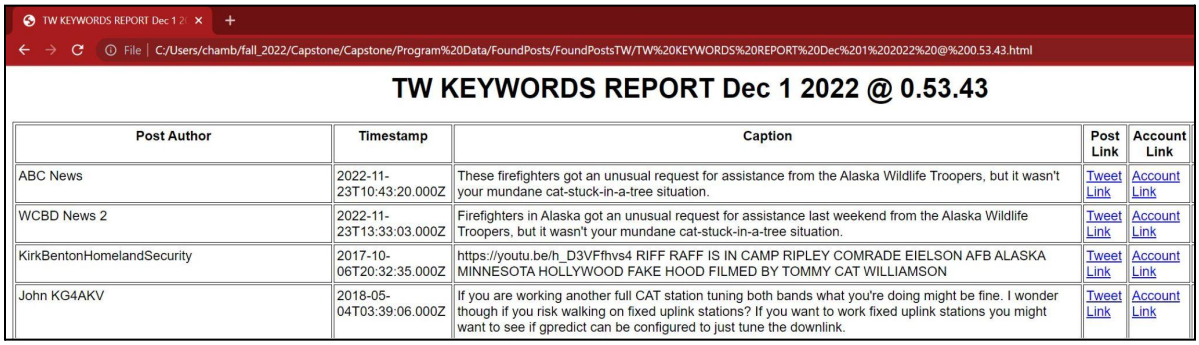
https://www.facebook.com/search/posts?q=**keyword**&filters=eyJycF9sb2NhdGlvbjowIjoie1wibmFtZVwiOlwibG9jYXRpb25cIixcImFyZ3NcIjpcIjEwNjE5OTYxOTQxMTA5MVwifSIsInJlY2VudF9wb3N0czowIjoie1wibmFtZVwiOlwicmVjZW50X3Bvc3RzXCIsXCJhcmdzXCI6XCJcIn0ifQ%3D%3D

The filter's value is constant for each location and that allows us to send requests looking for all keywords and users by replacing the keyword but retaining the filters. The Facebook scraper collects the post author, the location the post was tagged to, the caption of the post, a copy of single image media, and creates links to the author account and post in question. After looking for keywords, the Facebook scraper iterates through each user in the flagged users list and collects 20 scrolls worth of posts should their page contain that many, which for the general user is a substantial time period and parses these posts for the same information as the keywords protocol. Note that this does mean that for each scan the Facebook scraper does write two output files, one for Keywords and one for Flagged Users.

To retrieve information from Twitter, looking for the word “keyword”, near Anchorage (within 20 miles), and “Latest” turned on, the program would send the following requests:

https://twitter.com/search?q=**keyword**%20near%3AAnchorage%2C%20Alaska&src=typed_query&f=live

Any posts tagged under these crafted urls meet the criteria as the filtering is done through url filters, so posts found under these urls, if any, are parsed for relevant data. The Twitter scraper collects the Tweet author, the timestamp, the caption, a link to the Tweet and to the Tweet author’s profile, and does support downloading and displaying single image media, however this is much less common with Twitter. Twitter then iterates over the flagged users for Twitter in the same fashion as Facebook and also writes two reports per scan. Facebook and Twitter take an average of 45 minutes to scan all the location urls.



TW KEYWORDS REPORT Dec 1 2022 @ 0.53.43

Post Author	Timestamp	Caption	Post Link	Account Link
ABC News	2022-11-23T10:43:20.000Z	These firefighters got an unusual request for assistance from the Alaska Wildlife Troopers, but it wasn't your mundane cat-stuck-in-a-tree situation.	Tweet Link	Account Link
WCBD News 2	2022-11-23T13:33:03.000Z	Firefighters in Alaska got an unusual request for assistance last weekend from the Alaska Wildlife Troopers, but it wasn't your mundane cat-stuck-in-a-tree situation.	Tweet Link	Account Link
KirkBentonHomelandSecurity	2017-10-06T20:32:35.000Z	https://youtu.be/h_D3VFfhvs4 RIFF RAFF IS IN CAMP RIPLEY COMRADE EIELSON AFB ALASKA MINNESOTA HOLLYWOOD FAKE HOOD FILMED BY TOMMY CAT WILLIAMSON	Tweet Link	Account Link
John KG4AKV	2018-05-04T03:39:06.000Z	If you are working another full CAT station tuning both bands what you're doing might be fine. I wonder though if you risk walking on fixed uplink stations? If you want to work fixed uplink stations you might want to see if gpredict can be configured to just tune the downlink.	Tweet Link	Account Link

Figure 5, Example of a keywords scan of Twitter

6.9 Local Storage

In our initial implementation, we planned on writing results files into the FoundPosts directory (see below) as html documents. As we finished development of the Minimum Viable Product (MVP) in time, we expanded on that allowing the use of an external harddrive. Once the flagged posts have been parsed according to the above protocols, an html document of the scan is written to the directory specified by the user in the configuration panel. This may be the default directory, ./ProgramData/FoundPosts/, or an external drive. Relative paths are written into the media section as all images for each site are written to a common folder to reduce duplication, and the images path is rendered to display the image in the user's browser when viewing the report. Using this style of saving full report information allows all images to be stored in one location and using an html table makes the output easy to quickly understand and share. Html documents are an accepted format currently in use by our client. The files are named according to the time stamp the scan was run and the site scanned. Below is an example of a scan of Instagram rendered in the user's browser:

Figure 6, Example of results from an Instagram Scan

6.10 File System Hierarchy

The program is designed to be installed into the Program Files directory of a Windows Machine. In the top level directory, the executable of the program and a directory for Program Data are stored. Program data stores wordlists, images, results files if the default directory is used, region data, and user configurations according to the diagram (below).

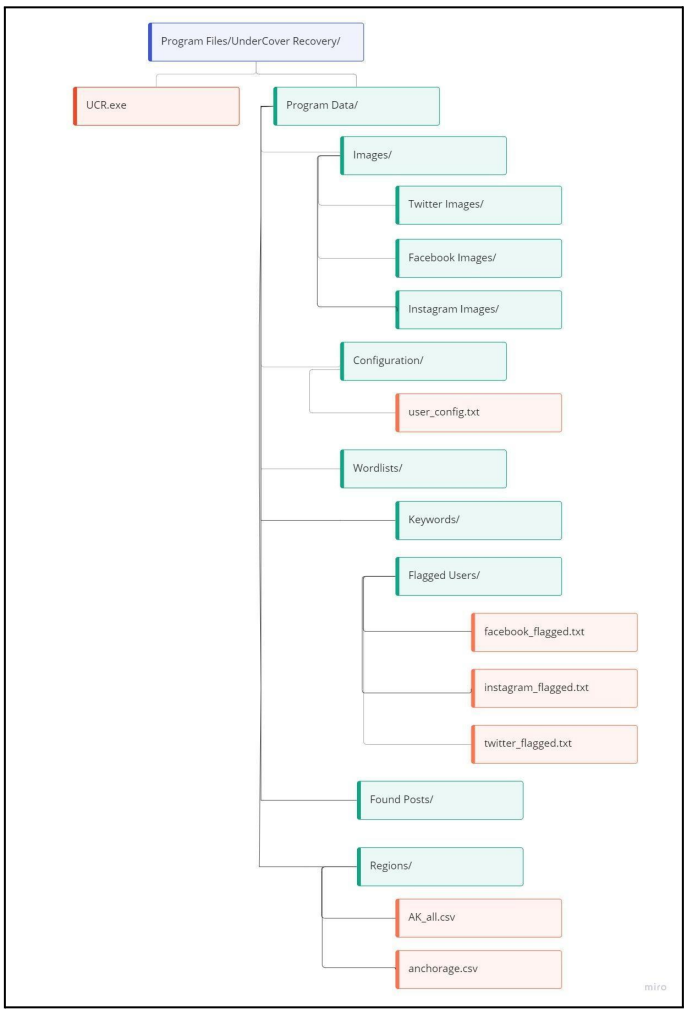


Figure 7, File System Hierarchy

7. Software Development

We used a prototyping model to first reach a MVP and then improve the design based on feedback from the client. We allotted time to learn the technologies, implement the design, test, and revise. We spent roughly a month implementing the MVP and using the remaining time expanding the functionality to include the other target sites and improve the aesthetics of the GUI.

7.1 Project Management Software

Pivotal Tracker was utilized for project management. The Epics we established were as follows in chronological order: Barebones GUI, File System, Web Scraping, Local Storage, Beautify GUI, and Results Formatting. We used Github for version control and Miro for diagramming.

7.2 Planning and Timeline

Given that we were unfamiliar with some of the required technologies, we allotted time to get up to speed, meet with our client, and establish requirements. After that, we spent the majority of our time in the implementation phase, as the project has several modular components. We had a functional barebones GUI operational in October, had the initial implementation of Instagram and Facebook scraping done by the beginning of November, and implemented Twitter scraping and GUI beautification through the first week of December.

7.3 Testing

We tested the design holistically and in its modularity. For the GUI we tested invalid inputs, premature program termination, and invalid wordlists. To test the web scraping functionality, we created social media accounts and made posts with specific terminology and verified the program can find and save those posts. The ultimate testing was turning prototypes of the product over to the client and getting their feedback once we reached the MVP threshold, and the client wanted some additional functionality we were able to implement in the form of advanced search operands for Instagram scraping.

8. Results

The project resulted in a fully functional Win32 application, meeting and exceeding the clients specifications. The GUI allows users to configure and select social media sites to scrape content from and the modular design of the application allows for the addition of new python scripts for new social media sites. The content can be saved to an external harddrive in addition to the local directory path, addressing the potential memory concerns of downloading copies of media to the local device.

The original approach would have used the Requests python library to scrape the html source code from each website, but Facebook and Twitter both place their html content into hashed values, making the source code unreadable when pulled via this library. To address this issue, the Selenium library was incorporated instead. It emulates the web browser and it mimics the manual process a user would go through to see content on these sites, allowing users to render the html source code outside of its hash. At that point the content can be grabbed and analyzed. This library was also used when the requests used in the process of getting cookies from the Instagram site changed, allowing us to continue on the project with minimal delays.

Some improvements can be made to the program, such as updating the permissions of the launch button of the GUI. Currently it will allow new threads of the program to spawn each time the Launch button is pressed even if a scan is currently underway, allowing for multiple scrapes to occur simultaneously. Allowing the user to launch unlimited threads in this way may cause the computer to freeze and should be discontinued. An attempt to mitigate this potential program was made through making the Launch button red and the text changes to Running, giving the button the appearance of being unclickable, however in our next iteration of the program this button would become literally unclickable. The program could also be made to adjust the GUI proportions to allow the window to be resized, but the current static pixel window creation makes it so resizing the window does not resize the layout or any child elements, so we have removed the ability to resize the window as a temporary fix. Additionally, this

program could be made to work on additional platforms or hosted as a website to allow universal access with internet connection, but security concerns would have to be addressed in its design.

9. Conclusion

The most difficult part of the project is the design, mainly due to the difficulty in creatively solving problems and achieving desired outcomes. Some of our designs were dropped as our client specified certain conditions to meet, such as our incorporation of proxy addresses. Our project also dealt with third party software in the form of each social media site, which we had to work around and was often designed to discourage website scraping in the first place. This consumed a great amount of project work time, as each site had to be analyzed and experimented on to scrape the correct content, and, in the case of Facebook and Twitter, locally rendered so as to get past the content hashing in the html source code. If there is a main takeaway from this project's development, it should be that the technology available today allows for any data on the internet to be captured en masse and quickly. Our tool has the potential to automate a time-consuming process for our client, but also serves as a reminder to the public to consider the volume of data made readily available through public postings.