Product: S.S. Media

Client: Doctor Judith E. Rosenbaum

BLACK BEAR ANALYTICS

Final Project Report

Black Bear Analytics

Abdullah Karim | Colleen DeMaris | Griffin Fluet | James West | Ryan Handlon

Revision History

Version Number	Release Date	Description
Version 1.0	05/07/2021	Original Release

Acknowledgements

Black Bear Analytics would like to thank Doctor Judith E. Rosenbaum for giving us the opportunity to work on this project. She has been a pleasure to work with on this project and we are excitedly waiting for the future comings of the S.S Media Scraper. We'd like to thank the Clement and Linda McGillicuddy Humanities Center for approving Dr. Rosenbaum's grant proposal, and Kristen Libby for helping us manage that funding. We would also like to extend our thanks to Professor Terry Yoo for helping us connect to these wonderful clients.

Table of Contents

S. S. Media Abstract

Social media has grown to be an extremely large part of society in recent years. Because of this, a need to gather data from it has also arisen. Black Bear Analytics is creating a tool that allows researchers to scrape public data from posts on Twitter and Instagram. This tool was requested by Doctor Judith Rosenbaum of UMaine's Department of Communication and Journalism. The S.S. Media will have an easy-to-navigate interface that allows users to switch between scraping public posts on Instagram and Twitter. The *New Search* page gives the user an option of either an advanced search or a basic search. The basic option searches by specified hashtags, locations, or phrases, and an acceptable start and end date to check with each post scraped, while an advanced search has the same general functions as the basic search, but adds the ability to search for more than one topic (hashtag, location, phrase), as well as run the search once every specified amount of time. The tool will store a number of scrape requests, so that users can either inspect a previous scrape request or request it again. Once the scraping is completed, the data will be downloaded to the user's computer in the form of a .csv file, and images will be stored in a corresponding folder. The outcome of this project will hopefully be that researchers like Doctor Rosenbaum can have a tool that will enable them to easily search for data and trends on social media sites.

0. Introduction

Black Bear Analytics is a University of Maine Computer Science capstone team composed of Abdullah Karim, Colleen DeMaris, Griffin Fluet, James West, and Ryan Handlon. Our client was Doctor Judith Rosenbaum, an associate professor of Communication and Journalism at UMaine.

0.1 Purpose

The purpose of this document is to provide an overview of the project now that it's been completed. It provides details about what the project was and what got completed. It also has documents created during development as attachments for further information. This includes the Software Requirements Specification Document (SRS), System Design Document (SDD), User Interface Design Document (UIDD), Code Inspection Review (CIR), User Manual (UM), and Administrator Manual (AM).

0.2 Intended Readership

The intended readership of this document is for anybody who wishes to learn about the S.S. Media Social Scraper and its development. This includes readers such as the client, future developers, future employers, and the capstone professor.

1. Executive Summary

Black Bear Analytics' client, Dr. Judith E. Rosenbaum, is an Associate Professor and Chair of the Department of Communication and Journalism. As such, she has a need to easily scrape and compile social media posts off of the internet for her research. Black Bear Analytics was chosen to solve this problem. After careful analysis of the requirements following design standards and methodologies, Black Bear Analytics has designed the S. S. Media Social Scraper. It is a web based application that allows users to log into a website, select specific search criteria, and scrape posts off of various Social Media sites such as Twitter and Instagram. Together, we feel this design meets all the needs and requirements of the client and can feasibly be implemented given the provided time constraints and funding. Details about methodologies, design, testing practices, and more are provided further along in this project report.

2. The Problem

The need for a social media scraping platform came about because of our client's, Dr. Judith E. Rosenbaum, research. A lot of Dr. Rosenbaum and her graduate students' research involves the analysis of social media posts. This includes notable examples such as #boycottnike and research into the acknowledgement of danger in posts at national parks. The hope for this project is to provide a tool that allows for easy gathering of social media posts to make this research easier, as the current systems allowing users to gather information from online social media sites are heavily fragmented and hard for individuals who aren't technically literate to use due to the requirement of learning how to program. Specifically, the UMaine Communications and

Journalism department experiences this issue when it comes to gathering information for their research.

3. Requirements

The main requirement that had to be completed was #6: Scrape Social Media. This goes hand in hand with requirement #7: Download Scraped Data. These two requirements were the heart of the project, as the goal is to provide Dr. Rosenbaum with the ability to download public posts for her research. Three were also other requirements that added features to make requirements #6 and #7 easier. This includes things like inputting and saving search criteria, as well as various account management requirements like log in, ban account, and approve account. For the full list of requirements, please refer to the BlackBearAnalytics_SRS document. Having all requirements implemented and working according to specifications indicates the project is done.

4. Initial Design

4.1 Features

Our customer requested a straightforward and easy to understand design, in both layout and functionality. She also wanted the software to have user handling, with individual accounts for each person to log into. Before logging in, the user would be able to either click the Contact us button to contact an administrator of the software, or click the Create Account button to request a new account for themselves. The Home page would show the user the previous five scrapes that they started. Once on the Home page, the user would be able to either enter the Admin Settings or the Basic Settings page, depending on their account permission level. The user would also be able to go to the Searching page, select Instagram or Twitter and fill out the appropriate searching boxes, as well as select an advanced search option that would give the user more power over specifically what they search for. Once the scrape is complete, the data would be immediately downloaded to the user's computer.

4.2 Back End

Originally, we and our client were looking at several different social media platforms to scrape. These included Facebook, Instagram, Twitter, Reddit, and TikTok. After some research and brief testing with the sites listed above, our client took our feedback and recommendations and narrowed the number of social media platforms to only Instagram and Twitter. Originally, our team was going to implement a semi-generic process for both Twitter and Instagram. URLs would be extracted and would be stored on a text file. After the links were stored, HTML would be extracted using a tool like Beautiful Soup to create a DOM. The resulting DOMs would then be placed onto a queue. A multithreading implementation would then begin to extract from the queue, saving any data that would match the given search criteria.

4.3 Hosting

In regards to hosting the project, our team started out with Amazon Web services, using an Elastic Cloud instance and a database to try to host the currently deployed software and the user database. The code itself has been hosted on GitHub for source control.

4.4 Front End

The design for the front end was created on a website resource called Figma, where the pages layouts were designed and saved. The team used ReactJS to start creating the JavaScript, HTML, and CSS required to build the pages. We had planned, and put in the Figma design, to implement popup text boxes to help the users at every point during the use of the program. These text boxes would pop up when a user hovered over a label and describe what the component does, and how it is used to interact with the rest of the program. The original design plans included making the styling dynamic enough to accommodate even mobile use, with shifting components based on the width and height of the screen.

5. Design Results

5.1 Features

The main features that were wanted, user friendliness and the scraping functionality, were implemented as planned. The JavaScript and HTML/CSS was written based on the Figma design that was created. Users are able to request a new account and log into an existing account as planned with the Login Page, but users cannot contact an administrator through the system, they will have to contact them in a separate manner. The Home page does not display previous searches. The user, from the Home page, can access their settings and their Searching page. Their settings page is different depending on whether the user is an admin or not, giving admins access to approving/banning/deleting/making an account an admin. The user would also be able to go to the Searching page, select Instagram or Twitter and fill out the appropriate searching boxes, but cannot select an advanced setting option for extra features. Once the scraping is done, the data is downloaded to the user's computer.

5.2 Back End

The overall architecture of the backend changed from what was planned in the beginning to what was actually implemented. The first thing that changed was the fact that we gained funding from the McGillicuddy foundation. This allowed us to look into options that would cost money, such as the Twitter API. This made scraping the tweets much simpler. The team also found a library that handles finding data on Instagram posts. The back end also added a new step called ScrapeHelper, that handles the setup of the scrapes for both processes.

5.3 Hosting

The software is implemented on a Raspberry Pi that is port-forwarded to give access to users. This Pi will be based in the research lab of Dr. Rosenbaum. It will contain the entirety of the software, as well as a backup in case something goes wrong. The code is still contained on GitHub for source control.

5.4 Front End

The design for the front end was pretty much followed from the Figma, including color choices, font styles, and spacing. Popup information boxes were not created for additional information. The team used ReactJS as planned to create and implement the JavaScript and HTML/CSS, however, implementing styling for mobile screens was not completed for all the pages.

6. Next in Line

6.1 Unfinished Business

There are several planned features that were not able to be implemented. The first, and likely the simplest, is the Contact Us page. This would be a button on the login page called “Contact Us” that redirects the user to a page with text fields to input their email address and the message they wish to send. The message field would be limited to 500 characters. On clicking submit, the main administrator of the software would receive an email to their specified account with the user’s message in the body of it.

The next feature is the saving of the search criteria. With this feature, the information of the previous five scrapes of each user would be saved to their profile in the database. This list of scrapes would be displayed on the user’s Home page, with the option to click them to run that one same search again.

The overall layout and design of the React components and CSS styling needs more polishing, on several aspects. Different browsers produce different results in the styles: this would need to be researched more and implemented so it is consistent. Different screen sizes also result in different changes to the styling, so the CSS must be edited to accommodate many different screen sizes.

6.2 Expanding the Software

Future work on this project could yield a lot of expansion. There is room to add more social media platforms to scrape, like Facebook, Reddit, even TikTok.

For administrators, a screen with a list of the scrapes that are running and what users are running what scrapes would be useful. This screen would also allow the administrator to shut down any scrapes that they deem no longer needing to go. The goal would be that if the administrator shut down the scrape, the process would exit as if it ended normally and simply download the found information to the user’s computer.

Automatically approving a user could also be on the list, similar to many other website’s approval process. The user inputs their preferred email, and the system sends that address an email for the user to click confirm on. This process is currently manually done through the Admin Settings page.

6.3 Testing

This project did not get as much attention to testing as our team would have liked. The plan was to use Selenium and Pytest to automate testing on both the backend and the front end. While the team placed coding conventions in place and did several manual tests, automated testing needed more work.

7. Future Reference

7.1 Dependencies

This project requires several dependencies to operate. These dependencies will be sorted by whether they are needed for the frontend, the backend,deployment, or for testing.

Frontend

npm:

- React
- React-Router
- react-confirm-alert
- file-saver
- jszip

Backend

pip:

- tweepy
- insta-scrape
- Flask-SQLAlchemy
- Flask
- selenium
- requests
- jsonify

Testing

selenium

Deployment

Docker - Docker-compose - docker aws cli image (version 2.0.6)

pip:

- pyopenssl

8. Summary and Conclusions

8.1 Summary

The S.S. Media was created for Dr. Judith Rosenbaum in an effort to make her data collection for her research easier. This project was created using several different tools, but the main ones have been Python, React, the Twitter API. There have been several issues that the team has run into with developing the S.S. Media, causing some of the features to not be completed as planned. However, the main purpose of the project has been completed successfully. The software runs a scrape as requested on Twitter or Instagram, and downloads the finished data to the user's computer.

8.2 Conclusion

There is still much work that can be done to polish S.S. Media and features to implement. The project was designed with many extra features to create the best product possible. BlackBearAnalytics was able to deliver much of the core functionality and produce a product that can greatly benefit Dr. Judith Rosenbaum's efforts in researching social media. Abdul, Colleen, Griffin, Ryan, and James would like to again thank the University of Maine, Professor Yoo, and Dr. Rosenbaum for an amazing experience. Our hopes are for another successful capstone team to take on this project and finish what we have started.

Appendix A.1 - Team Review Sign-off

All members of Black Bear Analytics have gone through the review process for this document and agree on both the manual entries in this document and are certain there is enough coverage to be effective. We have also received a sign-off from the client, approving this document which means they have reviewed and have accepted what is listed above until fixed.

Name	Signature	Date
Ryan M. Handlon	[illegible]	May 7th, 2021
Comments:		
Abdullah I. Karim	[illegible]	May 7th, 2021
Comments:		
Griffin L. Fluet	Griffin L. Fluet	May 7th, 2021
Comments:		
Colleen DeMaris	Colleen DeMaris	May 7th, 2021
Comments:		
James West	[illegible]	May 7th, 2021

Appendix B.1 - Document Contributions

Abdullah, Colleen, Griffin, James, and Ryan all contributed equally (20%) to this document. Each member added to a section(s) of the document with valuable content. In drafting this document each member helped contribute details based on their roles in the project, ensuring that any new readers get as much information as they need.

Appendix A - SRS

Product: S.S. Media

Client: Doctor Judith E. Rosenbaum

BLACK BEAR
ANALYTICS

Black Bear Analytics

Abdullah Karim | Colleen DeMaris | Griffin Fluet | James West | Ryan Handlon

Revision History

Version Number	Release Date	Description
Version 1.0	10/24/2020	Original Release

1. Introduction

The purpose of this section is to introduce the reader to both this document and the system requirements for the project, detailing introductory information that will be useful for the reader to know.

1.1 Purpose of this Document

The purpose of the SRS is to clearly outline the features agreed upon by both our client (Doctor Rosenbaum) and Blackbear Analytics for the S.S. Media scraping tool. This document will cover the scope of the product and detail all requirements of the application. This includes functional and non-functional requirements, user interface design, required deliverables, and open issues.

The intended readership of this document includes Developers, Testers, the Client, and End Users. Developers will review this document to more fully understand the product, what must be developed, and help guide their efforts in its creation. Testers can use this document to help guide them in their testing of the application helping guarantee proper functionality of the product. The Client will read this document to have a greater understanding of the product they commissioned as well as guarantee that their development team understands what they need out of the product. The end user would review this document to better understand the product's functionality and intended purpose.

1.2 References

This document does not include any cited information and does not require any references.

1.3 Purpose of the Product

The client for this product is Associate Professor and Chair of the Department of Communication and Journalism Doctor Judith E. Rosenbaum at the University of Maine in Orono. A great deal of hers and her grad student's research involves the analysis of social media and how people use it. Two notable examples of this are the Boycott Nike hashtag from late 2018 or current research on the analysis of people's perceived risk when taking pictures at national parks, and whether they acknowledge it. An important aspect of this research is data collection, which requires a way to scrape and compile information based on specific hashtags, times, phrases, or location from social media sites. Doctor Rosenbaum has had tools in the past that would do this data scraping for her, however those tools were complex and eventually became outdated. She now needs a new up to date scraper that is easy for the end user to understand.

1.4 Product Scope

The scope of this project is to provide a web based tool that can scrape information off of social media websites. The application will have a user-friendly design allowing them to login and scrape information from any supported social media platform. Scrapeable websites shall include Twitter and Instagram exclusively. Should time permit, scraping of Reddit, Facebook, Snapchat, and TikTok will also be implemented in that order. The user will be able to scrape for information based off of hashtags, dates, locations, and key words or phrases. The tool will then compile the information into an easily manageable format and provided to the end user. In the base version of the scraping tool, the application will not provide any analysis or manipulation of information after it has been gathered. As can be seen in Figure 1, the system will provide all the functionality required to manage user accounts, login, select scraping specifications, scrape data, and downloading data. Outside actors of the system will include: Researcher, Administrator, and Unauthorized User, the social media platforms being scraped which include but are not limited to Twitter and Instagram, and the storage location of scraped data.

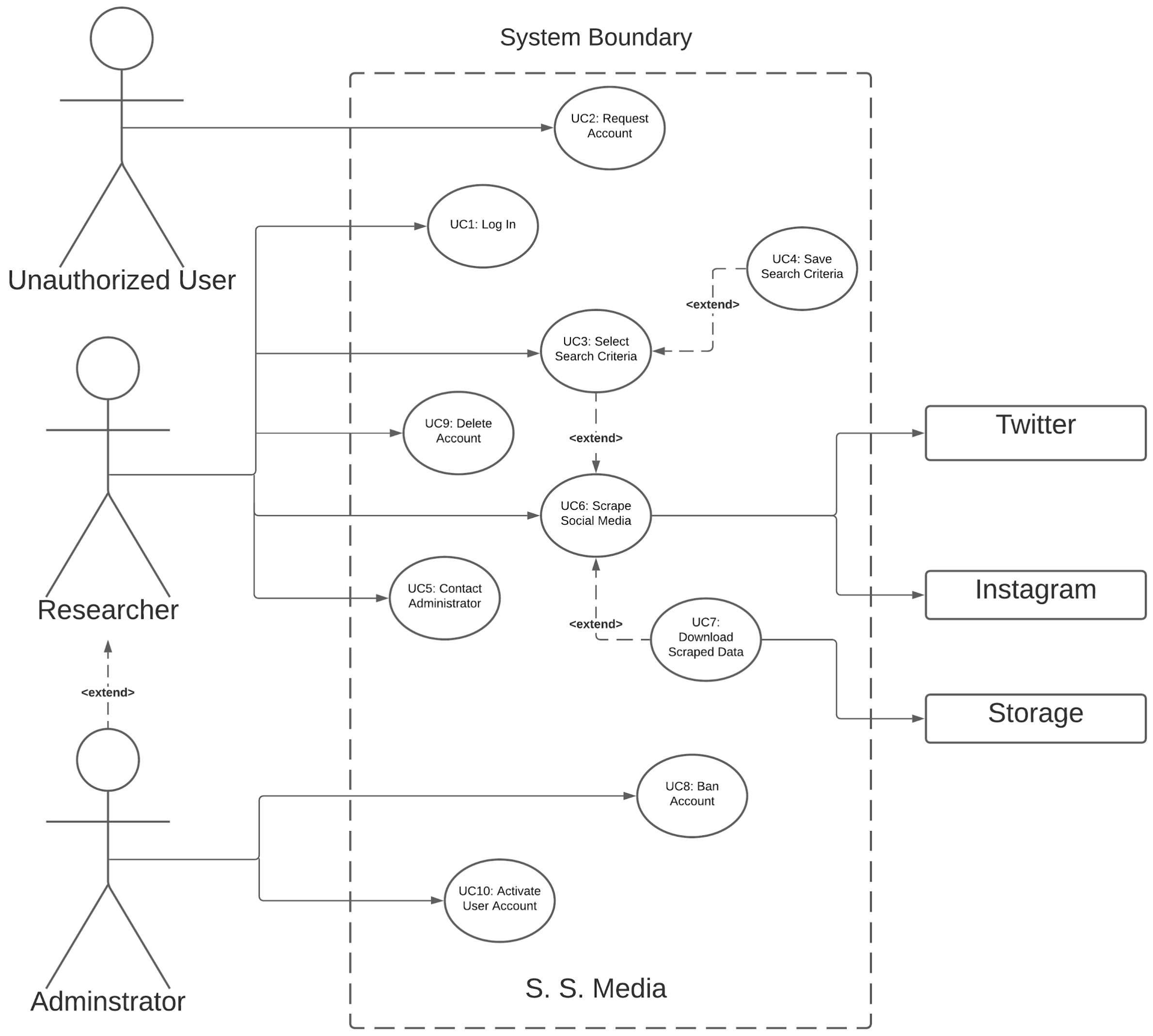


Figure 1: Use Case/Context Diagram of the S.S. Media Application

2. Functional Requirements

This section contains use cases that outline the functionality of our program. Per use case, there is a corresponding acceptance test titled in the use case description. The steps for each use case is written out at the end of the section for Functional Requirements. Functional Requirements are adapted from elicited user stories written while talking with the client about their needs. Use cases for open issues relation to functionality have no corresponding use case, but will be included at the time of closing the issue. Post minimum viable product (MVP) functionality is not included in this iteration of the SRS as they're open issues that are undergoing scope definition. For acceptance tests related to each functional requirement, refer to the list of tests in section 3.2 Acceptance Tests.

2.1 Use Case Descriptions

Number	1	
Name	Log In	
Summary	This use case addresses security for our system. Users must log in with a username and password combination that's given to them by an administrator upon request and approval of an account.	
Priority	4	
Preconditions	The user must be on the login screen. They must also have valid credentials.	
Postconditions	Users will be logged into the Home Page which displays their scrape history, settings, and a button that allows for the setup of a scrape.	
Primary Actor	Researcher	
Secondary Actors	Administrator	
Trigger	User enters the login screen	
Main Scenario	**Step**	**Action**
	1	User clicks username field. System activates the name field and allows the user to type into it.
	2	User enters a username. The system saves the information the user has entered.
	3	User clicks the password field. System activates the password field and allows the user to type into it.
	4	User enters password. As the password is being typed, the system displays it as dots to the user while also saving their input for authorization.
	5	System makes the greyed out login button active and blue so the user can log in.
	6	User clicks the login button. If the user's credentials are valid, they are logged in.
Extensions	**Step**	**Branching Action**
	6a	User failed authentication: Else, red text appears above the username field stating:"Invalid credentials. Please try again or contact the administrator."
Open Issues	Refer to Open Issue 2	

Number	2
Name	Request Account
Summary	Users request an account by clicking a button and filling out/sending a form for administrator approval.
Priority	4

Preconditions	User needs an account	
Postconditions	User submits a form for the user account that's sent to the administrator.	
Primary Actor	Researcher	
Secondary Actors	Administrator	
Trigger	Click the Request Account button from the Login window.	
Main Scenario	**Step**	**Action**
	1	User navigates to the Login window and clicks the request account.
	2	System produces a form with details of Email, First Name, Last Name, and Password text fields.
	3	User fills out the form and clicks the Request Account button.
	4	User is prompted to verify their email and the system sends the verification email with instructions.
	5	User opens the email and clicks the link to the verification page.
	6	System verifies the user.
	7	User is prompted that their account is verified and awaiting approval from an administrator. The system adds the new user account to the Administrator's Approve Accounts list.
Extensions	**Step**	**Branching Action**
	3a	The user does not enter a valid password and is not meeting the systems password criteria. The user is prompted to enter a password with at least 1 uppercase and 1 lowercase letter, 1 special character, 1 number, and be at least 8 characters long.
Open Issues	Refer to Open Issue 2	

Number	3	
Name	Select Search Criteria	
Summary	User selects their search criteria such as hashtags, locations, platform being scraped, and date range.	
Priority	5	
Preconditions	User must be logged in on the home screen.	
Postconditions	Users have a set of search criteria that they can then use to scrape data and/or save the search criteria.	
Primary Actor	Researcher	
Secondary Actors	Administrator	
Trigger	User clicks the New Search button or clicks a saved previous search.	
Main Scenario	**Step**	**Action**
	1	User clicks the New Search button.
	2	User selects the social media platform they are scraping.
	3	System loads search criteria for specified social media.
	4	User inputs basic search criteria into associated boxes. (Hashtags, Locations, Phrases)
Extensions	**Step**	**Branching Action**
	1a	User selects a Saved Search criteria from a list of saved criteria on the Home Page..
	1a1	System loads Scrap Search Criteria page and populates it with saved criteria. Skip steps 2-4.
	3a	User selects Advanced Search

	4a1	User inputs advanced search criteria (Hashtags, Locations, Phrases, Date Range, and/or logic, etc.)
Open Issues	Refer to Open Issue 1	

Number	4	
Name	Save Search Criteria	
Summary	Users can save searched criteria for future use.	
Priority	3	
Preconditions	Users are on the Social Platform Selection screen and have selected search criteria.	
Postconditions	The search criteria of a user is saved with their information.	
Primary Actor	Researcher	
Secondary Actors	Administrator	
Trigger	Save Search Criteria button from the Social Platform Selection screen.	
Main Scenario	**Step**	**Action**
	1	User clicks the Save Search Criteria button.
	2	System creates a popup to prompt for name of the search criteria.
	3	User enters a name in a popup prompt.
	4	System saves that search criteria to the scrape history associated with the user's profile in the user database.
Extensions	**Step**	**Branching Action - N/A**
Open Issues	N/A	

Number	5	
Name	Contact Administrator	
Summary	Users having trouble logging in or who have general issues with the software can click the Contact Administrator button on the login window and submit a form to administrators so issues can be resolved.	
Priority	1	
Preconditions	User is on the Login page.	
Postconditions	User submits a form with the submit button, sending their contact information and a description of the issue to admins.	
Primary Actor	Researcher	
Secondary Actors	Administrator	
Trigger	Click the Contact Administrator Button.	
Main Scenario	**Step**	**Action**
	1	User clicks the Contact Administrator Button.
	2	System provides a form where the user can enter their contact information and description of their issue.
	3	User enters contact information and description of their issues.
	4	User clicks the submit button.
	5	System sends this information to Administrator accounts.
Extensions	**Step**	**Branching Action**
	1a	User clicks the cancel button to retract their contract form.
Open Issues	N/A	

Number	6
Name	Scrape Social Media

Summary	The system will initiate a search for all relevant information from the selected social media platform.	
Priority	5	
Preconditions	User is on the Social Platform Selection screen and has filled in all relevant search criteria.	
Postconditions	Data relating to the search criteria will be gathered from relevant social media sites.	
Primary Actor	Researcher	
Secondary Actors	Administrator	
Trigger	Click the Perform Scrape button	
Main Scenario	**Step**	**Action**
	1	User clicks the Start Scrape button.
	2	System accesses selected social media platform.
	3	System runs algorithm scraping information with an input of selected criteria.
	4	System finishes running algorithm and prompts user to download scraped data. (see Download Scraped Data Use case).
Extensions	**Step**	**Branching Action**
	4a	User clicks the Halt Scrape button.
	4a1	System stops the scraping algorithm and returns the user to the home screen, throwing out any captured data in the process.
Open Issues	Refer to Open Issue 3	

Number	7	
Name	Download Scraped Data	
Summary	Once a scrape is finished, the data is downloaded to the user's local machine so they are accessible to the user.	
Priority	4	
Preconditions	User has hit the Scrape button and a scrape has been performed.	
Postconditions	All information associated with a scrape is stored out to excel files and folders on the user's desktop.	
Primary Actor	Researcher	
Secondary Actors	Administrator	
Trigger	Scrape has completed.	
Main Scenario	**Step**	**Action**
	1	The system has met the search criteria's time constraints, and finishes searching.
	2	The system prompts the user if they want to download or discard the scraped data.
	3	User selects download
	4	The system downloads the data to the user's device and saves it to the Downloads folder.
Extensions	**Step**	**Branching Action**
	3a	User selects Discard.
	3a1	The system deletes the scraped data and returns the user to the home page. Skip step 4
Open Issues	Refer to Open Issue 3	

Number	8

Name	Ban Account	
Summary	Administrators are allowed to ban user accounts that exist.	
Priority	1	
Preconditions	User must be an admin logged in.	
Postconditions	A user account is added to a blacklist.	
Primary Actor	Administrator	
Secondary Actors	N/A	
Trigger	User selects an account from the Edit User Accounts list and clicks the Ban button.	
Main Scenario	**Step**	**Action**
	1	User clicks a checkbox next to a user’s name.
	2	User clicks the Ban button..
	3	The system displays a verification popup.
	4	User clicks the Yes I Am Sure button.
	5	The system removes the account from the user database.
Extensions	**Step**	**Branching Action**
	1a	User clicks the Select All checkbox.
	1a1	User clicks the Ban button.
	1a2	The System returns to the Edit User Accounts page.
Open Issues	N/A	

Number	9	
Name	Delete Account	
Summary	Researchers are allowed to delete user accounts that exist.	
Priority	1	
Preconditions	User must be a logged in researcher. User must be on the settings page.	
Postconditions	A user account is deleted.	
Primary Actor	Researcher	
Secondary Actors	N/A	
Trigger	User clicks the delete account button.	
Main Scenario	**Step**	**Action**
	1	User navigates to the settings page.
	2	User clicks the Delete Account button.
	3	The system displays a verification popup.
	4	User clicks the Yes I Am Sure button.
	5	The system removes the account from the user database.
Extensions	**Step**	**Branching Action**
	4a	User selects the “No, I am not sure" button.
	4a1	The system returns the settings page.
Open Issues	N/A	

Number	10
Name	Activate User Account
Summary	Administrators must approve user accounts when a form request is received. Approving a user account gives them access to the website beyond the login page.
Priority	2
Preconditions	Must be logged in as an Administrator and on the Edit User Accounts screen.

Postconditions	A user’s verified account is activated, giving them access to the tools and functionality.	
Primary Actor	Administrator	
Secondary Actors	Unauthorized User	
Trigger	Account request form is submitted from an Unauthorized User.	
Main Scenario	**Step**	**Action**
	1	Administrator clicks the checkbox next to the new user.
		System displays information about the new user.
	2	Administrator clicks the Approve button.
	3	New user removed from the Approve Accounts list and gains researcher permissions.
Extensions	**Step**	**Branching Action**
	2a	Administrator clicks Deny button.
	2a1	New User is informed they have been denied via email, account is deleted, and removed from the Approve Accounts list.
Open Issues	N/A	

3. Non-Functional Requirements

This section of the document will detail all Non-Functional requirements for this project. All Non-Functional Requirements will be accompanied by a unique id number, a priority, a clear description, and a test that will be used to verify the requirement has been met.

3.1 Non-Functional Requirements

ID	Description	Priority	Tests
1	The system shall authenticate (or prevent authentication) to users within 5 seconds of the user clicking the login button.	5	Authorization Test
2	The system shall save the search criteria successfully 99% of the time when the user clicks the "Save Search Criteria" button.	4	Search Feature Test Scraping Test
3	The system shall return scraped, text-based information in a CSV file 99% of the time when the user clicks the start scrape button.	2	Scraping Test
4	The system shall return scraped, visual-based information 99% of the time when the user clicks the start scrape button	2	Search Feature Test Scraping Test
5	The system shall return scraped information within an overall time period of no more than 1 seconds per post being scraped	3	Scraping Test

6	The system shall load the saved search criteria successfully 99% of the time	5	Search Feature Test Scraping Test
7	The system shall provide accurately scraped data with 98% confidence.	5	Data Accuracy Test
8	The system shall be Open Source.	1	Licensing Test
9	The system shall have 99% of the information in it's user database encrypted.	4	Encryption Tests
10	The system shall be compatible with popular platforms including Google Chrome, Firefox, Safari, and Microsoft Edge.	2	Compatibility Test

3.2 Acceptance Tests

Authorization Test
Corresponding Non-Functional Requirement ID's: 1
Corresponding Functional Requirement ID's:

Enter username and password. Start a timer and then click the Login button. When the user is logged in, stop the timer. If the timer is greater than 5 seconds the test fails, otherwise it passes.

Search Feature Test
Corresponding Non-Functional Requirement ID's: 2, 4, 6
Corresponding Functional Requirement Use Cases: 3

Go to a "@socialmediaplatform". Post a "@testdata" on the developer account being used with a keyword to test such as "@parameter" with a unique id that goes with the post so that it's easily identifiable post-scrape. Navigate to the social platform selection screen on the web scraper. Enter selection criteria such as "@parameter". Run the scrape and see if the information provided contains the unique id specified. Repeat per @socialmediaplatform, @testdata, and @parameter. If it succeeds 99% of the time, the test passes. Else, test fails.

Scraping Test
Corresponding Non-Functional Requirement ID's: 2, 3, 4, 5, 6
Corresponding Functional Requirement Use Cases: 6, 7

Navigate to the Social Platform Selection screen on the web scraper. Enter selection criteria. Start a timer and hit the scrape button simultaneously. Keep track of the amount of requests. Once the process is done, stop the timer and check for any data produced in the downloads folder. If there is data and the total time taken for the scrape divided by the number of scrapes is under 1 second, the test succeeded. Else, test failed.

Data Accuracy Test
Corresponding Non-Functional Requirement ID's: 7

Scrape information from a social media platform. If there was an image, access the image and see if it's good quality. If it is, use the image's filename as a unique id to check the excel sheet for any corresponding text information. If there is, verify that information and the image corresponds to the post on the social media platform that it's describing. If the information is just text (such as a tweet), go to that text's corresponding post on the social media platform and verify that the information is the same. If it succeeds 99% of the time, the test passes. Else, test fails.

Licensing Test
Corresponding Non-Functional Requirement ID's: 8

Open the softwares licensing agreement. Read it. If the license states the software is Open Source the test passes, otherwise it fails.

Compatibility Test
Corresponding Non-Functional Requirement ID's: 10

Open the web scraper in "@webbrowser". Run all relevant tests to the application including the Authorization Test, the Search Feature Test, the Scraping Test, the Data Integrity Test, and the testing of basic website functionality. If all tests pass, the compatibility test passes for "@webbrowser". Repeat test for all applicable browsers.

Encryption Test
Corresponding Non-Functional Requirement ID's: 9
Corresponding Functional Requirement Use Cases: 2, 10

Open the application and create a new account with test user information. On the administrator account, approve the account request. Log in as the new user. Request test user information from the user database. Compare the raw data received to the actual user data. If they do not match, the received data looks encrypted, and isn't in plain text, the test passes. Otherwise the test fails.

Login Test
Corresponding Functional Requirement Use Cases: 1

Get an approved account from through the administrative process. Attempt to log into the platform with fake login credentials. The system will give you an error and tell you to try again. Enter the correct credentials for a user account. The system brings you to the Homepage of the scraper tool within 5 seconds of clicking the Login button. If the system did not give an error or the correct credentials failed to log the user in, this test fails. Else, this test passes.

Save Search Test
Corresponding Functional Requirement Use Cases: 4

Log into the scraper platform with valid credentials. Select the New Search Button. Enter search criteria and a platform for scraping. Click the Save Search Criteria button. Fill out the form that asks for a name of the search and hit submit. Navigate to the Home Page where saved search criteria names are shown and alongside the New Search button. If the search criteria name that was saved is there, click it (Else, the test fails). If the Social Platform Selection screen is brought up and the saved search criteria are autofilled, this test passes. Else, test fails.

Administration Test
Corresponding Functional Requirement Use Cases: 5, 8, 9

Open the application and create a new account with test user information. Log in to the administrator account and approve the new test account request. Log in as the new test user and fill out the Contact Administrator form with sample test information. Go back to the administrator account and check for a notification with the corresponding test data. If it's not there, the test fails, otherwise continue. Ban the new test account. Try to log in to the test account. If the login is successful, the test fails, otherwise continue. On the administrator account, unban the test account. Try to log in to the test account. If login fails, the test fails otherwise continue. On the test account, navigate to the settings page. Click the Delete Account button and select yes on the Are you sure prompt. After being redirected to the login page, try to login to the test account. If the login is successful, the test fails. Else, this test passes.

4. User Interface

See "User Interface Design Document" for *S.S. Media*. This document is a deliverable due on November 22nd, 2020 and will be prototyped, with a final version not available until then.

5. Deliverables

This section of the document provides a list of all deliverable items throughout the course of this project. Each deliverable will be accompanied by information about it's due date, what format it is to be delivered in, method of delivery, and any notes.

5.1 Deliverable Schedule

Name of Deliverable	**Format of Deliverable, Method of Delivery**	**Date of Delivery to Client**	**Notes**
Budget Estimate	.xlsx (Excel), through email and GitHub	October 14th, 2020	
Systems Requirement Specification	PDF, through email and GitHub	October 23rd, 2020	At this time Doctor Rosenbaum can look over the document and suggest any changes before signing it.
System Design Document	PDF, through email and GitHub	November 9th, 2020	
User Interface Design Document	PDF, through email and GitHub	November 22nd, 2020	
User Manual	PDF, through email and GitHub	April/May, 2021	
Administrator Manual	PDF, through email and GitHub	April/May, 2021	
Update Manual	PDF, through email and GitHub	April/May, 2021	This manual is used to find places in the code (in addition to code comments) that need to be updated if social media platforms update their APIs so that future developers can still use this program once platforms have changed. It will also outline how to use the program for new users as well as have a changelog of information
Executable Program	.src files, website link through email and GitHub	April/May, 2021	

6. Open Issues

The Open Issues section is a list of all currently known problems or questions pertaining to software requirements that need answering. Each open issue will be accompanied by a brief explanation of the issue, a timeline for when the issue will be resolved, and a specification of who is in charge of solving it.

Open Issue #1: Search Criteria

It is unknown whether the specified search criteria (Hashtags, Locations, Phrases) is viable for all social media platforms being scraped. For example, Twitter uses geolocation tags while Instagram does not all the time. There may also be special search criteria that are specific to individual social media. These search criteria need to be researched and a client meeting about the topics needs to happen. This issue shall be resolved within 20 days of the signing of this document and is assigned to Developer Griffin Fluet.

Open Issue #2: Number of Users

The amount of supported users at any given time is yet to be determined. Currently, we go with one user who carries the role of Administrator and Researcher, but we'd like to see if this can be expanded to support more than one user at a time. This criteria needs to be researched and a client meeting needs to happen to resolve the situation. This issue shall be resolved within 14 days of the signing of this document and is assigned to Developer Colleen DeMaris.

Open Issue #3: Size of Files

How much information that can be collected in a single scraping session and the potential size of that file are potential problems. Allowing for scrape sessions and file sizes that are too big could result in a download or transfer that could break something. This criteria needs to be researched and consulted on with the client. This issue shall be resolved within 14 days of signing this document and is assigned to Developer James West.

Appendix A – Agreement Between Customer and Contractor

The client (Doctor Rosenbaum) and Blackbear Analytics agree that all requirements in this document will be implemented in good faith. All requirements will be implemented as close to the specifications as possible, with margins for error being slim or none. In the event that implementation must diverge from the specifications, the development team will notify the client as soon as possible and work together with the client until a suitable agreement is reached. If the client would like to add requirements after signing this document, Blackbear Analytics bears no responsibility for the failure to implement new requirements, but will attempt to add them in good faith. Should any new requirements be added, the deliverables schedule in section 5 of this document will be updated immediately to reflect changes, and the requirements discussed will be reflected in this document by the next deliverable deadline and changes will be finalized with the client.

	Name	Signature	Date
Customer:			
Comments:			
Team:			
	Ryan M. Handlon	Ryan M. Handlon	October 24rd, 2020
	Abdullah I. Karim	Abdullah I. Karim	October 24rd, 2020
	Griffin L. Fluet	Griffin L. Fluet	October 24rd, 2020
	Colleen DeMaris	Colleen DeMaris	October 24rd, 2020
	James West	James West	October 24rd, 2020

Appendix B – Team Review Sign-off

By signing your name below, you acknowledge that you are a member of Blackbear Analytics and have read the document with a solid comprehension of the scribed materials. You agree to complete all requirements stated on this document as is, in good faith. Should any new requirements come up, you agree to assist in re-drafting this document for approval as described in Appendix A. You agree that it is not required to complete requirements added after this version is released, but understand that all requirements, new or old, must be implemented in good faith.

Name	Signature	Date
Ryan M. Handlon	Ryan M. Handlon	October 24rd, 2020

Comments:

Name	Signature	Date
Abdullah I. Karim	Abdullah I. Karim	October 24rd, 2020

Comments:

Name	Signature	Date
Griffin L. Fluet	Griffin L. Fluet	October 24rd, 2020

Comments:

Name	Signature	Date
Colleen DeMaris	Colleen DeMaris	October 24rd, 2020

Comments:

Name	Signature	Date
James West	James West	October 24rd, 2020

Comments:

Appendix C – Document Contributions

It’s worth mentioning that while each section varied in size, each team member did preliminary research that went into the decision-making process for much of this document which is a major factor considered when creating Appendix C.

Each member contributed to drafting this document evenly. Ryan Handlon created the Introduction (Section 1) and did much of the review for the document, earning a 25% share. Abdullah Karim contributed to the Functional Requirements (Section 2) and the review of the document with a percentage contribution of 25%. Colleen DeMaris wrote the Deliverables (Section 5) section of this document with a percentage contribution of 20%. Griffin created the Non-Functional Requirements (Section 3) section and had a 15% contribution to the document. James wrote out Section 4, Open Issues, Appendix A, and Appendix B which led to a 15% contribution. Blackbear Analytics as a team contributed to Appendix C to ensure transparency and talk through team dynamics.

Appendix B - SDD

Product: S.S. Media

Client: Doctor Judith E. Rosenbaum

System Design Document

Black Bear Analytics

Abdullah Karim | Colleen DeMaris | Griffin Fluet | James West | Ryan Handlon

November 10, 2020

Revision History

Version Number	Release Date	Description
Version 1.0	11/10/2020	Original Release

1. Introduction

The purpose of this section is to introduce the reader to both this document and the system architecture for the project, detailing introductory information that will be useful for the reader to know.

1.1 Purpose of This Document

The purpose of the Software Design Document (SDD) is to clearly outline the system design agreed upon by both the client (Doctor Rosenbaum) and Blackbear Analytics for the S.S. Media scraping tool. The Software Design will meet all requirements specified in the Software Requirements Specification (SRS). This document will cover all system design details. This includes an Architectural Design Diagram (ADD), a Technology Architecture Diagram (TAD), a Decomposition Description, Database Descriptions, File Descriptions, and a Requirements Matrix.

The intended readership of this document includes Developers, Testers, the Client, and End Users. Developers will review this document to more fully understand the system architecture and to help guide their efforts in the system's creation. Testers can use this document to help guide them in their testing of the application helping guarantee proper functionality of the product. The Client will read this document to have an understanding of the design of the product commissioned as well as guarantee that their development team understands what is needed out of the product. The end user would review this document to better understand the product's system design.

1.2 References

Banga, S. (2020). *What is Web Application Architecture? Components, Models, and Types*. Hackr.io. https://hackr.io/blog/web-application-architecture-definition-models-types-and-more.

Fulton, J. (2018, December 12). *Web Architecture 101*. https://engineering.videoblocks.com/web-architecture-101-a3224e126947.

Kerins, I. (2019, July 4). Solution Architecture Part 5: Designing A Well-Optimised Web Scraping Solution. https://blog.scrapinghub.com/solution-architecture-part-5-designing-a-solution-estimating-resource-requirements.

Wikimedia Foundation. (2020, October 29). *Web crawler*. Wikipedia. https://en.wikipedia.org/wiki/Web_crawler.

Other References:
Black Bear Analytics SRS Document

UI Design :
https://www.figma.com/file/qc0MoZOfnAJrwfYVoisxcN/BlackBearAnalytics_Mockup?node-id=0%3A1

AWS : https://aws.amazon.com/

2. System Architecture

The system architecture describes the flow, processing, and storage of information within the S.S. Media product. The architecture below includes an Architecture Design Diagram for a Scraper System as well as an accompanying Technology Architecture Diagram which illustrates the technology being used at each step in the design process. A Design Class Diagram is included in Section 2.2 which illustrates the object-oriented functionality of S.S. Media with descriptions of each step in the process.

2.1 Architectural Design

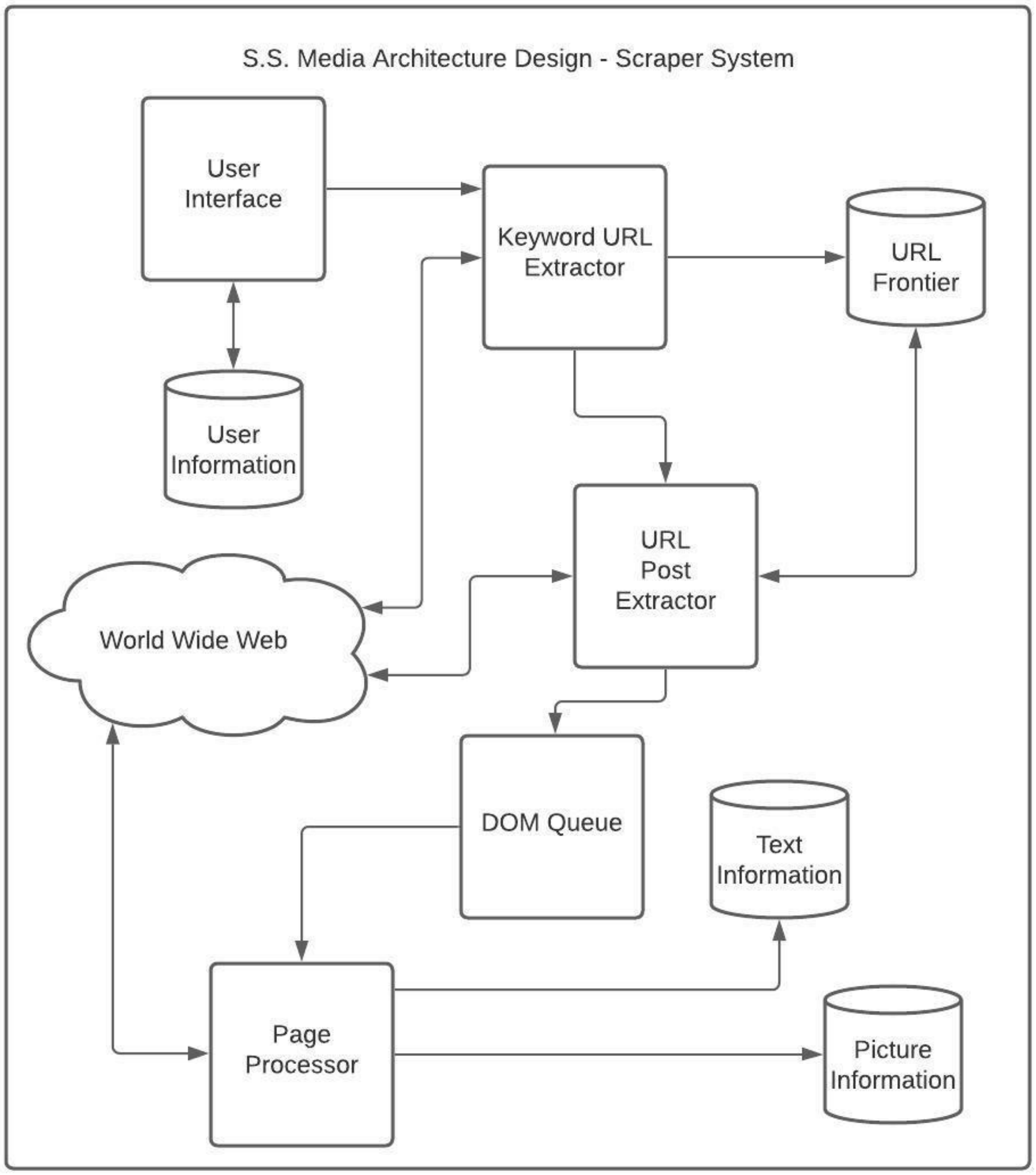


Figure 2.1: S.S. Media Architecture Design

Figure 2.1 shows the S.S. Media Architecture Design. As can be seen in the diagram, the Architecture starts with the User Interface, which manages user information and sends a requested scraped to the Keyword URL Extractor. The Keyword URL Extractor populates the URL Frontier, with URLs needed to be scrapped. From there the URL Post Extractor takes posts from the URL Frontier and creates Document Object Model (DOM) Objects which are added to the DOM Queue. The DOM Queue one by one sends DOMs to the Page Processor. There, the DOM's get parsed for relevant information and which is added to the text and picture information databases.

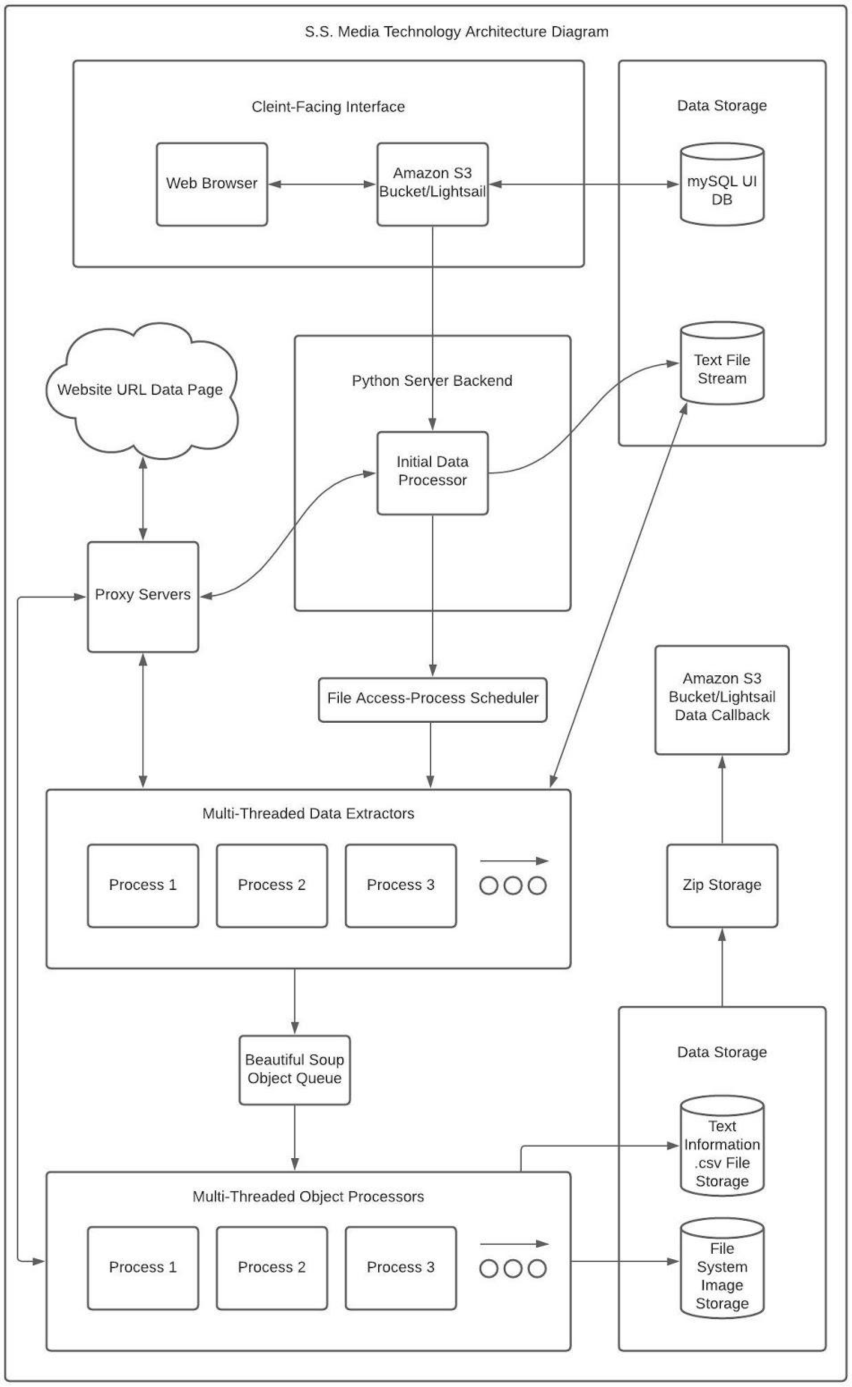

S.S. Media Technology Architecture Diagram
Cleint-Facing Interface
Web Browser
Amazon S3 Bucket/Lightsail
Data Storage
mySQL UI DB
Text File Stream
Website URL Data Page
Python Server Backend
Initial Data Processor
Proxy Servers
File Access-Process Scheduler
Amazon S3 Bucket/Lightsail Data Callback
Multi-Threaded Data Extractors
Process 1
Process 2
Process 3
Zip Storage
Beautiful Soup Object Queue
Data Storage
Text Information .csv File Storage
Multi-Threaded Object Processors
Process 1
Process 2
Process 3
File System Image Storage

Figure 2.2: S.S. Media Technology Architecture Diagram

Figure 2.2 depicts the specific technologies that play into each part of S.S. Media's design. The user interface a client sees is accessed through a browser hosted through Amazon Web Services (AWS). The AWS will be a Python Web App service (through the Lightsail service) that the system will use to create a page where information can be input and dynamically output. AWS also provides an S3 bucket service that allows for object storage which can be used for data storage at runtime.

Any user inputs flow through the system into a Python backend where Initial data processing occurs and the URL frontier is made into a text file which the application uses to get each post's information. Information then flows through Data Extractors which create the Beautiful Soup Objects (Document Object Models, or DOMs) and are Queued up for multiple processors to then grab and process information from. DOMs are a representation of a post that processors can use to extract post information. Note that as Processors gather information, the product connects to web pages via a proxy to maintain the integrity of the root IP should an IP ban be issued by a social media platform.

Information is then stored into .csv files if they are text and into an image file system otherwise. The storage units are zipped into a file that the client can access via the user interface.

2.2 Decomposition Description

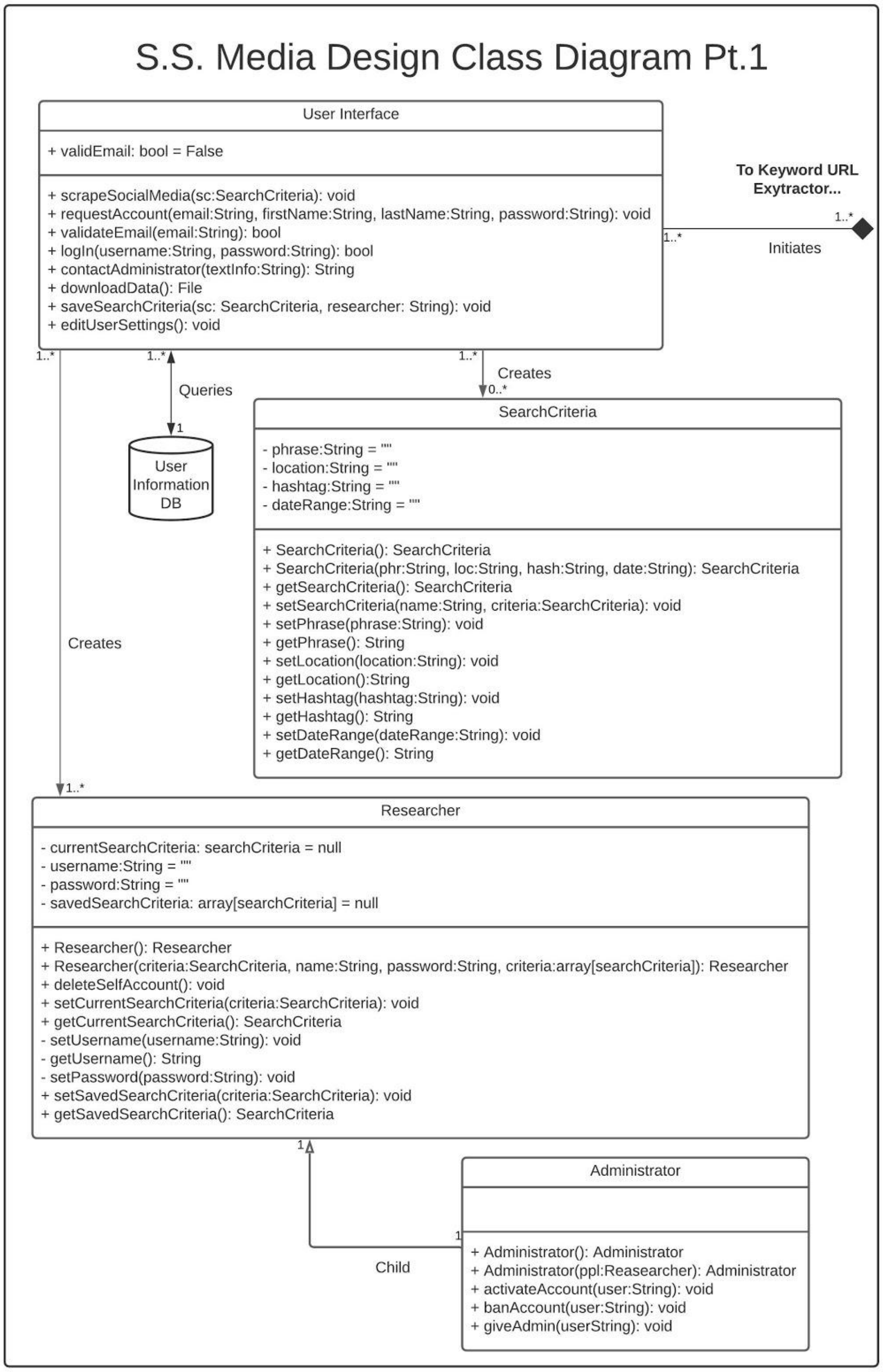

S.S. Media Design Class Diagram Pt.1
User Interface
+ validEmail: bool = False
+ scrapeSocialMedia(sc:SearchCriteria): void
+ requestAccount(email:String, firstName:String, lastName:String, password:String): void
+ validateEmail(email:String): bool
+ logIn(username:String, password:String): bool
+ contactAdministrator(textInfo:String): String
+ downloadData(): File
+ saveSearchCriteria(sc: SearchCriteria, researcher: String): void
+ editUserSettings(): void
To Keyword URL Exytractor...
1..*
1..*
Initiates
1..*
1..*
Queries
1
User Information DB
1..*
Creates
0..*
SearchCriteria
- phrase:String = ""
- location:String = ""
- hashtag:String = ""
- dateRange:String = ""
+ SearchCriteria(): SearchCriteria
+ SearchCriteria(phr:String, loc:String, hash:String, date:String): SearchCriteria
+ getSearchCriteria(): SearchCriteria
+ setSearchCriteria(name:String, criteria:SearchCriteria): void
+ setPhrase(phrase:String): void
+ getPhrase(): String
+ setLocation(location:String): void
+ getLocation():String
+ setHashtag(hashtag:String): void
+ getHashtag(): String
+ setDateRange(dateRange:String): void
+ getDateRange(): String
Creates
1..*
Researcher
- currentSearchCriteria: searchCriteria = null
- username:String = ""
- password:String = ""
- savedSearchCriteria: array[searchCriteria] = null
+ Researcher(): Researcher
+ Researcher(criteria:SearchCriteria, name:String, password:String, criteria:array[searchCriteria]): Researcher
+ deleteSelfAccount(): void
+ setCurrentSearchCriteria(criteria:SearchCriteria): void
+ getCurrentSearchCriteria(): SearchCriteria
- setUsername(username:String): void
- getUsername(): String
- setPassword(password:String): void
+ setSavedSearchCriteria(criteria:SearchCriteria): void
+ getSavedSearchCriteria(): SearchCriteria
1
1
Child
Administrator
+ Administrator(): Administrator
+ Administrator(ppl:Reasearcher): Administrator
+ activateAccount(user:String): void
+ banAccount(user:String): void
+ giveAdmin(userString): void

Figure 2.3: S.S. Media Design Class Diagram Part 1

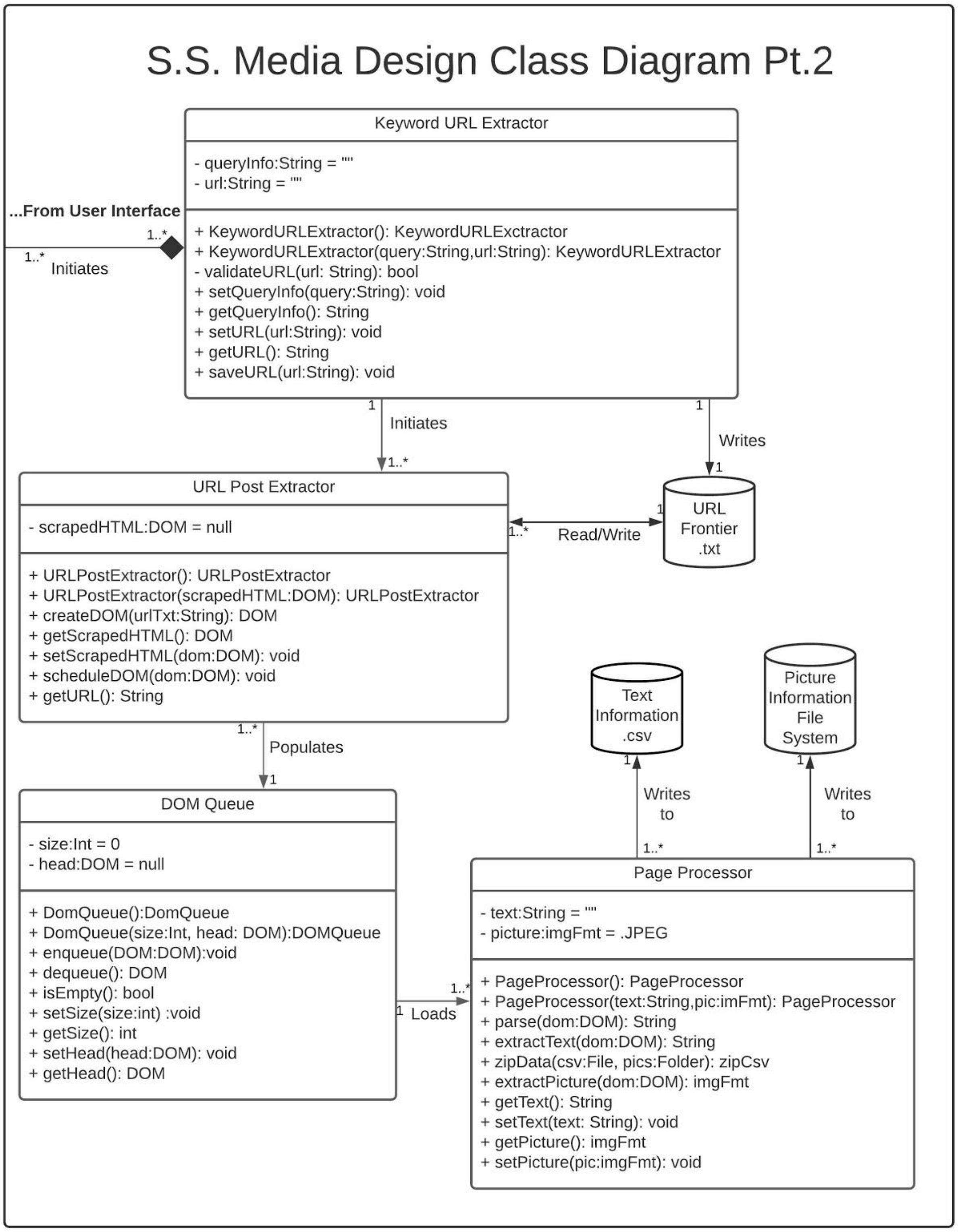

S.S. Media Design Class Diagram Pt.2
Keyword URL Extractor
- queryInfo:String = ""
- url:String = ""
+ KeywordURLExtractor(): KeywordURLExctractor
+ KeywordURLExtractor(query:String,url:String): KeywordURLExtractor
- validateURL(url: String): bool
+ setQueryInfo(query:String): void
+ getQueryInfo(): String
+ setURL(url:String): void
+ getURL(): String
+ saveURL(url:String): void
...From User Interface
1..*
1..*
Initiates
1
Initiates
1..*
1
Writes
1
URL Frontier .txt
1
1..*
Read/Write
URL Post Extractor
- scrapedHTML:DOM = null
+ URLPostExtractor(): URLPostExtractor
+ URLPostExtractor(scrapedHTML:DOM): URLPostExtractor
+ createDOM(urlTxt:String): DOM
+ getScrapedHTML(): DOM
+ setScrapedHTML(dom:DOM): void
+ scheduleDOM(dom:DOM): void
+ getURL(): String
1..*
Populates
1
Text Information .csv
Picture Information File System
1
Writes to
1..*
1
Writes to
1..*
DOM Queue
- size:Int = 0
- head:DOM = null
+ DomQueue():DomQueue
+ DomQueue(size:Int, head: DOM):DOMQueue
+ enqueue(DOM:DOM):void
+ dequeue(): DOM
+ isEmpty(): bool
+ setSize(size:int) :void
+ getSize(): int
+ setHead(head:DOM): void
+ getHead(): DOM
1
Loads
1..*
Page Processor
- text:String = ""
- picture:imgFmt = .JPEG
+ PageProcessor(): PageProcessor
+ PageProcessor(text:String,pic:imFmt): PageProcessor
+ parse(dom:DOM): String
+ extractText(dom:DOM): String
+ zipData(csv:File, pics:Folder): zipCsv
+ extractPicture(dom:DOM): imgFmt
+ getText(): String
+ setText(text: String): void
+ getPicture(): imgFmt
+ setPicture(pic:imgFmt): void

Figure 2.4: S.S. Media Design Class Diagram Part 2

Figures 2.3 and 2.4 show the planned S.S. Media Design Class Diagram (DCD). It has 11 main components which are: the User Interface, the Keyword URL Extractor, the URL Frontier file, the URL Post Extractor, the DOM Queue, the Page Processor, the Text Information .csv file , the Picture Information File System, the User Information Database, the Search Criteria Object, and the Researcher Object. The following paragraphs each describe a main component into more detail including information such as what their job is and how they will operate. The storage will not be included in this description as they are described above and/or in Section 3.

The User Interface will be the only thing the user will ever interact with. It's main purpose is to manage the user and their information as well as initiate the scraping process. It will allow the user to do tasks such as log in, contact an administrator, select search criteria, save search criteria, start a scrape, etc. The User interface will be able to store information such as usernames, passwords, saved scrape criteria, settings, etc. in a User Information Database. Upon the start of a scrape, the User Interface will send the user's search criteria to the Keyword URL Extractor for processing.

The Keyword URL Extractor will be used to handle the front lines of the S.S. Media's scraping. After receiving a set of search criteria from the User Interface, it's job is to create a URL Frontier, a text file of the URLs of each post to be scraped later. This will be done by creating a URL associated with the specified search criteria and scraping that URL's HTML. The Keyword URL Extractor then parses the HTML for links to posts that need to be scraped and validates those links. After validation, each link gets added to the URL Frontier Database.

The URL Post Extractor's job is to extract post HTML and send it to the DOM Queue (Document Object Model). The URL Post Extractor will iterate through the URL Frontier. For each URL, it will visit the social media post's web page, scrape it's HTML, build that HTML into a DOM Object, send the DOM Object to the DOM Queue, and remove the URL from the URL Frontier.

The DOM Queue's job is to manage DOM Objects while the URL Post Extractor and Page Processor run. As they run, it will receive DOM Object's from the URL Post Extractor and add them to the end of the Queue while concurrently sending DOM Objects at the head of the Queue to the Page Processor.

The Page Processor's job is to parse the DOM Objects. It will go through all information in the DOM Object and scrape the information that is requested. If this includes links to pictures, the Page Processor will download the pictures from the internet. The scraped information will be saved into a Text Information .csv file. The pictures will be saved to a Picture information folder with a unique ID number as the file name linking it to its corresponding text information.

The Search Criteria Object manages a user's specified search criteria. It will hold data values such as Hashtags, Phrases, and Locations. It will also include functions to manipulate those parameters, create Search Criteria Objects, and save search criteria for later use.

The Researcher Object will be used to manage user accounts. It will hold data values such as current search criteria, username, password, and saved search criteria. It will also include functions to manipulate those parameters, create a new Researcher Object, and delete the Researcher Object. The Researcher Object also has a child Administrator Object. The Administrator Object contains all functionality of the Researcher Object, but also includes functions to create an administrator account, approve user accounts, edit account names, ban accounts, and give administrator privileges to user accounts.

3. Persistent Data Design

The Persistent Data Design section details how system databases and files used by S.S. Media will be designed. Each database and file description will also include a diagram for further detail.

3.1 Database Descriptions

The database structure consists of one SQL database, containing {x} tables. The first table is the user table, with its purpose being to contain any information necessary in order to validate/invalidate a user when they try to log in. This table has columns for a name, username, password (hashed), type of account (admin, basic), and email address. The string variables will be stored in a VARCHAR format, which dynamically adjusts to the length of the string. The size of a VARCHAR is the length of the data stored, plus two bytes.

Table 3.1 - User Table Schema

Name of Field	**Data Type**	**Max Data Size**	**Description**
Primary Key: unique_ID	String: VARCHAR	22 bytes	The primary key of the table. Each user has a specific key randomly generated upon account creation.
first_name	String: VARCHAR	22 bytes	The first name of the user.
last_name	String: VARCHAR	22 bytes	The last name of the user.
hashed_password	String: VARCHAR	52 bytes	The password of the user, chosen by them and encrypted before being stored.
account_type	int	4 bytes	The type of account for the user. Basic (represented by 0) accounts can adjust personalization settings and can run search criteria and gain results. Admin (represented by 1) accounts can do everything that Basic accounts can, as well as assign other users as Admins.
Foreign Key: email_address	String: VARCHAR	52 bytes	The email address of the user, in order to validate the identity of the person if they forget their password or type it wrong too many times. This is also used for the user to log into their

			account, rather than a username.

The next table is dedicated to saved search criterias. The purpose of this table is to keep a record of all the searches that want to be saved. If the user wants to make a search happen once a day every day for a week, this is where the information gets stored in order to tell the program when to search. This table includes hashtags, locations, platform being scraped, and date range.

Table 3.2 - User Input Schema

Name of Field	**Data Type**	**Max Data Size**	**Description**
Primary Key: email_address	String: VARCHAR	52 bytes	The primary key of the table. Each search criteria has a specific key based upon a user's email address
hashtag(s)	String: VARCHAR	252 bytes	This is the list of hashtags in the saved search. They are separated by the hashtag symbol and are in a String format. For example, #nike#boycott#boycottnike
location(s)	String: VARCHAR	252 bytes	The locations of the places in the media being scraped
platform	String: VARCHAR	22 bytes	The specific platform to scrape, either Twitter or Instagram
date_start	timestamp	8 bytes	The start date of the search, as a timestamp
date_end	timestamp	8 bytes	The end date of the search. If no end date is specified, then the search is a one-time search
scrape_frequency	int	4 bytes	The frequency of the scrape between the start and end times.
scrape_interval	time	8 bytes	The amount of time

			between each scrape.

3.2 File Descriptions

The files used by the system include a JSON file, text file, a picture file, and a csv file. The JSON file is used to store the user configuration settings. These settings will be set up as labels with an associated text value. The settings text will be read into the system as Strings that vary in length or as a Boolean value, depending on the setting. The following include the configurable settings with the label and default text associated with the label.

Table 3.3 - Settings

Label	Data Type	Default Value	Size
download_location	String	C:\Downloads	256 bytes
view_scrape_history	Boolean	true	1-bit
advanced_search_default	Boolean	true	1-bit
email_notifications	Boolean	true	1-bit
email	String	User's initial sign-up email	256 bytes

The text file being used by the system is a file that will temporarily store URL links in a URL Frontier. This file will be used as an intermediary to pass data from our URL Extractor to our URL Post Extractor.

Table 3.4 - Text Data Format

URL	Data Type	Size
twitter.com/example1	String	512 bytes
twitter.com/example2	String	512 bytes

The Picture file will be used to collect and organize all pictures stored in a given scrape. The Picture file will be pushed as a download to the user's configured download location upon completion of a scrape. Each picture will be saved with a unique name corresponding to it's scraped text information in the .csv file.

Table 3.5 - Picture Data Format

Picture File	Data Type	Size
pigtureID#1919193.png	Varying (.png,.jpg,.jpeg,.gif,etc)	Up to 12mb
pigtureID#1903577.png	Varying	Up to 12mb

	(.png,.jpg,.jpeg,.gif,etc)	

The .csv file will be used as a final way to condense, sort, and organize the data collected. This .csv file will be pushed as a download to the user's configured download location upon completion of a scrape. The scraped data .csv file structure consists of n columns depending on the website.

Table 3.6 - Text File Structure

Twitter Headers	**Data Type**	**Size**	**Desc.**	**Instagram Headers**	**Data Type**	**Size**	**Desc.**
ID	String	128 bytes	The tweet's ID	ID	String	128 bytes	The instagram post's ID
Date	int	4	Date tweet was posted	Date	int	4	Date photo was posted
Time	int	4	Time tweet was posted	Time	int	4	Time photo was posted
Timezone	String	128 bytes	Timezone tweet was posted in	Timezone	String	128 bytes	Timezone photo was posted in
User_ID	String	256 bytes	User's 'handle' that posted the tweet	User_ID	String	256 bytes	User's 'handle' that posted the photo
Username	String	256 bytes	User's display name	Likes	int	16 bytes	Number of likes
Tweet	String	560 bytes	Tweet contents	Replies	String	512 bytes	Content of replies
Replies	String	560 bytes	Replies' content	Location	String	256 bytes	Location the user was at when posting the photo
Retweets	int	16 bytes	Number of times retweeted	Hashtags	String	256 bytes	Hashtags used in the description

Likes	int	16 bytes	Number of times liked	Link	String	512 bytes	Link to the post
Location	String	256 bytes	Location the user was at when the tweet was posted	Image_ID	UUID	128 bytes	An ID for the photo associated with the post
Hashtags	String	560 bytes	Hashtags used in the tweet	Description	String	512 bytes	Contents of description
Link	String	512 bytes	A link to the tweet				

4. Requirements Matrix

This section links the functionality described in the SRS document with the System Components and functionality designed in this SDD document. Below is a table that illustrates requirement-component connections. Components are taken directly from the Design Class Diagram (Figure 2.3 and 2.4), so looking at that for reference will help.

Table 4.1 - Requirement-Component Connection

Functional Requirements	**System Components -> Function**
Use Case #1: Log In	User Interface -> logIn(), Researcher -> getUsername(), Researcher -> setUsername(), Researcher -> getPassword(), Researcher -> setPassword()
Use Case #2: Request Account	User Interface -> contactAdministrator(), User Interface -> validateEmail(), Administrator -> approveAccount(), Administrator -> giveAdmin()
Use Case #3: Select Search Criteria	User Interface -> ScrapeSocialMedia(), Search Criteria -> All accessor/constructors/mutators()
Use Case #4: Save Search Criteria	User Interface -> saveSearchCriteria()
Use Case #5: Contact Administrator	User Interface -> contactAdministrator()
Use Case #6: Scrape Social Media	User Interface -> scrapeSocialMedia(), Keyword URL Extractor -> All functions(), URL Post Extractor -> All functions(), DOM Queue -> All functions(), Page Processor -> All functions()
Use Case # 7: Download Scraped Data	User Interface -> downloadData(), Page Processor -> downloadText(), Page Processor -> downloadPictures()
Use Case #8: Ban Account	Administrator -> banAccount()
Use Case #9: Delete Account	Researcher -> deleteSelfAccount()
Use Case #10: Activate User Account	Administrator -> activateAccount()

Appendix A – Agreement Between Customer and Contractor

The client (Doctor Rosenbaum) and Blackbear Analytics agree that the system's architecture meets all requirements signed in the SRS document and will be implemented in good faith. All designs will be implemented as close to the specifications in the system architecture as possible, with margins for error being slim or none. In the event that implementation must diverge from the design, the development team will notify the client as soon as possible and work together with the client until a suitable agreement is reached. If the client would like to add designs/requirements for the system's architecture after signing this document, Blackbear Analytics bears no responsibility for the failure to implement new requirements, but will attempt to add them in good faith. Should any new requirements be added, the deliverables schedule in section 2 and 3 of the SRS document will be updated immediately to reflect changes, and the designs/requirements discussed will be reflected in this document by the next deliverable deadline and changes will be finalized with the client.

Name	Signature	Date
Customer:		

Comments:

Team:

Name	Signature	Date
Ryan M. Handlon	[illegible]	November 8th, 2020
Abdullah I. Karim	[illegible]	November 8th, 2020
Griffin L. Fluet	Griffin L. Fluet	November 8th, 2020
Colleen DeMaris	Colleen DeMaris	November 8th, 2020
James West	James West	November 8th, 2020

Appendix B – Team Review Sign-off

By signing your name below, you acknowledge that you are a member of Blackbear Analytics and have read the document with an in-depth comprehension of the scribed materials. You agree to complete all designs stated on this document as is, in good faith. Should any new designs come up, you agree to assist in re-drafting this document for approval as described in Appendix A. You agree that it is not required to complete designs added after this version is released, but understand that all designs, new or old, must be implemented in good faith.

Ryan M. Handlon	[illegible]	November 8th, 2020
Abdullah I. Karim	[illegible]	November 8th, 2020
Griffin L. Fluet	Griffin L. Fluet	November 8th, 2020
Colleen DeMaris	Colleen DeMaris	November 8th, 2020
James West	James West	November 8th, 2020

Comments:

Appendix C – Document Contributions

Each member contributed to drafting this document evenly (20% each). Ryan Handlon created the Introduction (Section 1), aided in the creation of the System Architecture (Section 2.1), made the description for the Decomposition Description (Section 2.2), and helped in the revision process. Abdullah Karim helped in the creation of the Architectural Design Diagrams (Section 2.1), the development of the Decomposition Diagram (2.2), and the Requirements Matrix (Section 4). Griffin Fluet created the Design Class Diagram (Section 2.2), and helped in the revision process. Colleen DeMaris created the Persistent Data Design (Section 4) and helped in the revision process. James West also created the Persistent Data Design (Section 4) and helped in the revision process. The Appendices were appended and modified from the SRS document and Appendix C have been read and reviewed by the whole team.

Appendix C - UIDD

Product: S.S. Media

Client: Doctor Judith E. Rosenbaum

User Interface Design Document

BLACK BEAR
ANALYTICS

Black Bear Analytics

Abdullah Karim | Colleen DeMaris | Griffin Fluet | James West | Ryan Handlon
November 24, 2020

Revision History

Version Number	Release Date	Description
Version 1.0	11/24/2020	Original Release

1. Introduction

The purpose of this section is to introduce the reader to both this document and the system architecture for the project, detailing introductory information that will be useful for the reader to know.

1.1 Purpose of This Document

The purpose of the User Interface Design Document (UIDD) is to clearly outline the user interface design agreed upon by both the client (Doctor Rosenbaum) and Blackbear Analytics for the S.S. Media scraping tool. The User Interface will meet all requirements specified in the Software Requirements Specification (SRS). This document will cover all user interface design details. This includes User Interface Standards, a Navigation Diagram, a User Interface Walkthrough, Data Validation, and a Report Formats.

The intended readership of this document includes Developers, Testers, the Client, and End Users. Developers will review this document to more fully understand the system user interface and to help guide their efforts in the system's creation. Testers can use this document to help guide them in their testing of the interface helping guarantee proper functionality of the product. The Client will read this document to have an understanding of the design of the product commissioned as well as guarantee that their development team understands what is needed out of the product. The end user would review this document to better understand the product's user interface design.

1.2 References

Black Bear Analytics SRS Document

Black Bear Analytics SDD Document

UI Design :
https://www.figma.com/file/qc0MoZOfnAJrwfYVoisxcN/BlackBearAnalytics_Mockup?node-id=0%3A1

2. User Interface Standards

The User Interface Standards section will go over the design standards and themes that are used throughout the user interface. This section also touches on the general error handling.

As seen in Figure 2.1 and 2.2, the drop down boxes are of the same component and will display an example of input in the field when the user has not selected an option yet. Clickable buttons will only be made clickable when valid input in the corresponding text field(s) is entered, until then the button will be greyed out and not be able to be clicked. Once valid input is entered into every field needed the button will be highlighted and clickable. Toggles will be highlighted when clicked to represent that they are selected and greyed out when not. Labels for buttons and text fields will be descriptive as to what the button does when clicked or what is intended to be entered in the text field. Text fields also provide an example of possible input above the field. Navigation of the user interface will be discussed later in this document. All of the themes listed above are common components within the user interface and will be used throughout.

General error handling will typically attempt to send the user back to the last screen they were at with all of the valid inputs, if any, that the user had entered before the error occurred. If this is not possible due to corrupt or invalid data, the user will be sent back to the home page, and the user will need to manually enter valid input data again. For example, if the user were to hit an error during the Search Criteria stage, having entered the Platform and a single Hashtag they wished to look for, then the user would be sent back to the Search Criteria stage with the selected platform still selected, and the Hashtag still entered. The user will make another attempt to enter valid inputs, and continue on without further interruption. If the user's search criteria data, such as the selected platform and hashtag, were to be corrupted, the user would then be sent back to the home page, without any inputs previously entered being saved. The user would then need to go back to the Search Criteria page and manually re-enter any valid inputs they had before the error occurred.

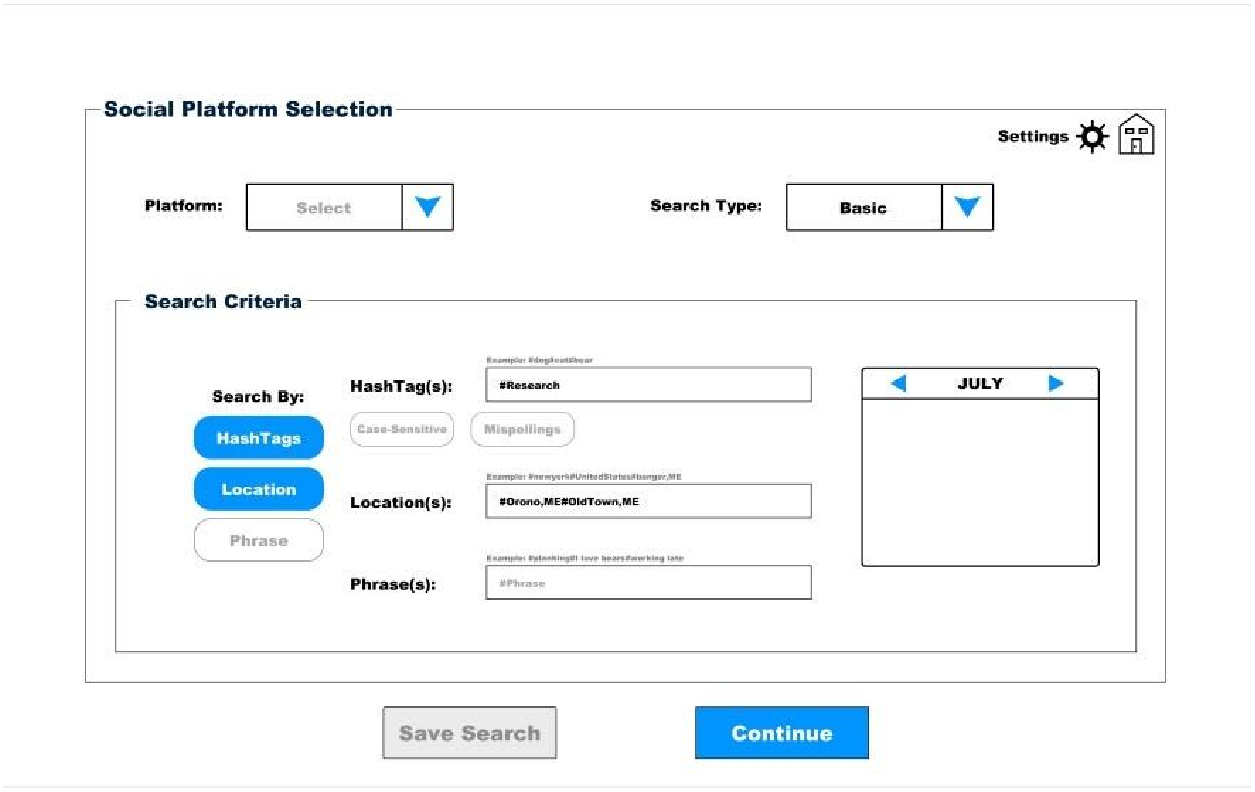


Figure 2.1: New Search Page

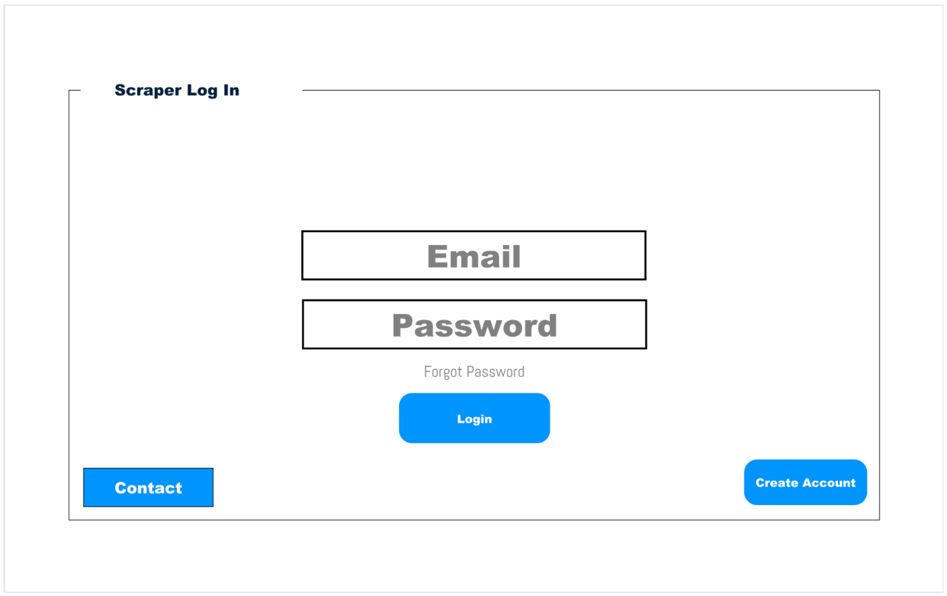


Figure 2.2: Login Page

3. User Interface Walkthrough

The User Interface Walkthrough section is intended to help visualize what the website will look like. It will include a Navigation Diagram detailing which pages can navigate to which. Following that will be a walkthrough of each unique page on the website including information on how to navigate to and from the page, what the purpose of the page is, it's main functions, and descriptions of the functionality of all buttons and fields on the page.

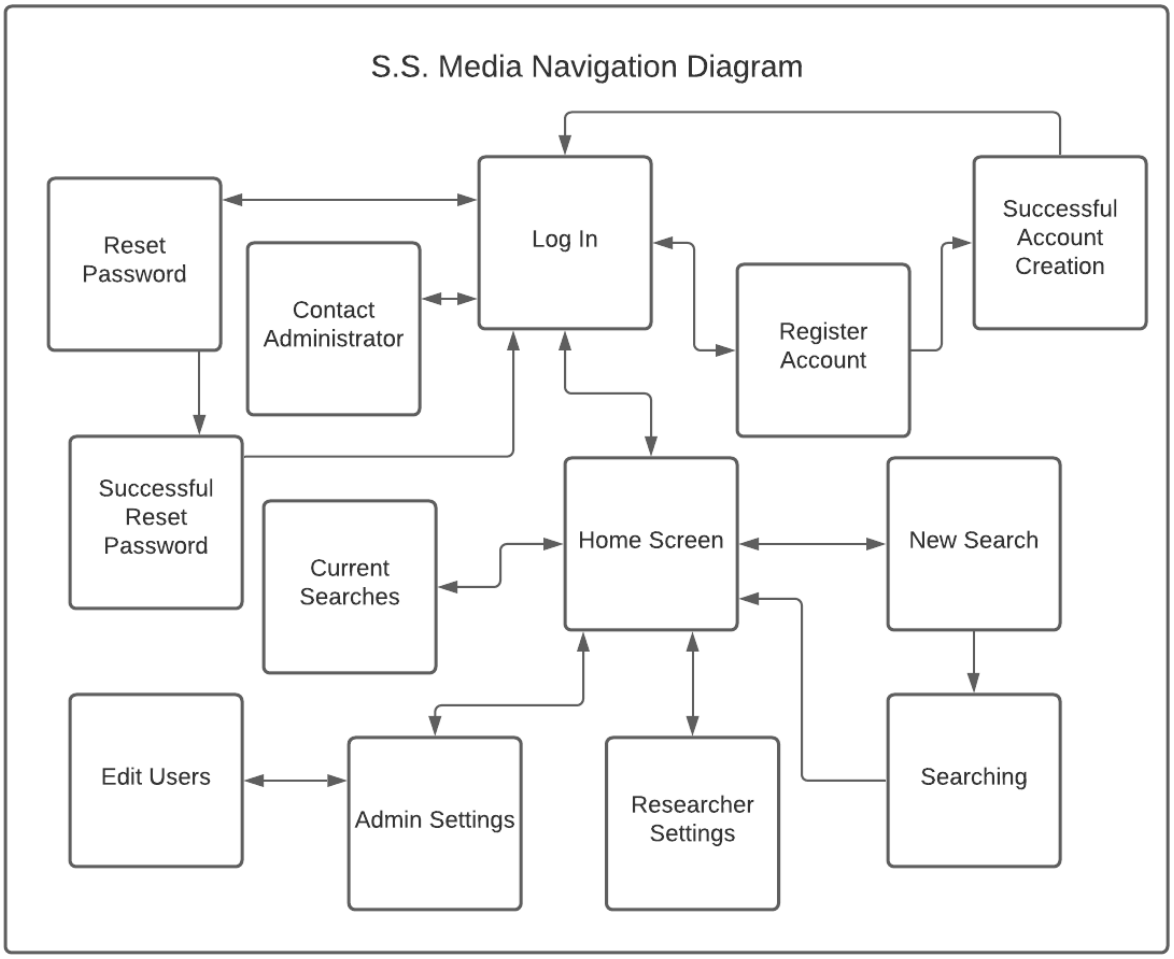


Figure 3.1: S.S. Media Navigation Diagram

The S.S. Media Navigation Diagram, as seen in Figure 3.1, displays all pages that can be visited by the user on the scraping website. When a user navigates to the website for the first time, they will be brought to the login page. From there they will be able to navigate to any other page by the way of buttons. The Navigation Diagram shows all relationships between the pages. An arrow pointing from one page to another indicates that there is a way to navigate to the pointed at page from the current page. The following page descriptions will show a mockup of each page, including how to navigate to and from the page, what the purpose of the page is, it's main functions, and descriptions of the functionality of all buttons and fields on the page.

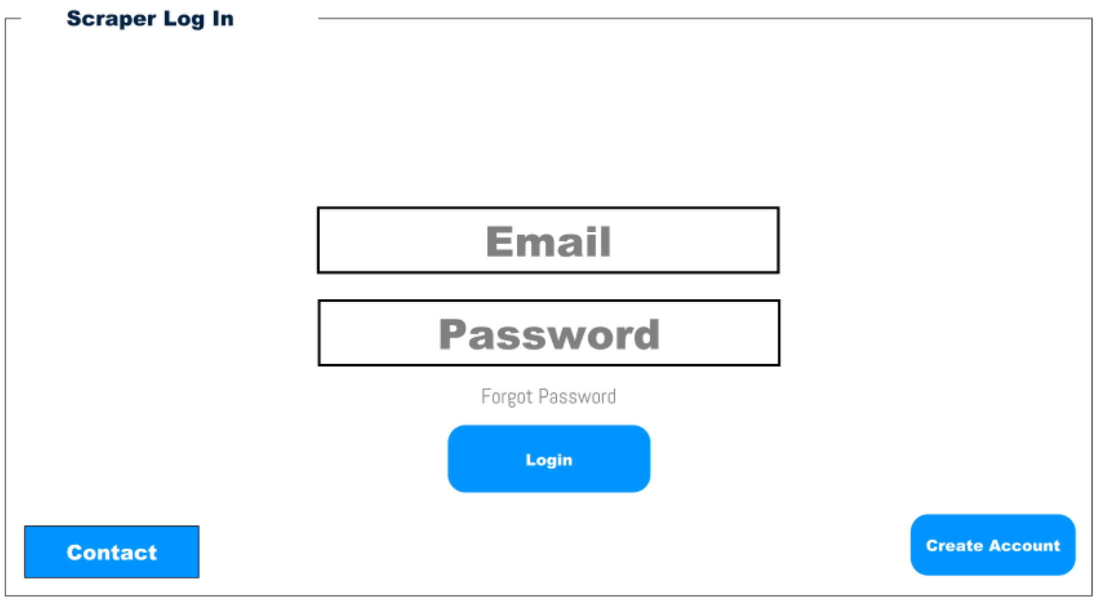


Figure 3.2: Login Page

The Login Page, Figure 3.2, is the first screen a user will see upon entering the site. It is navigated to either as the first page a user would view upon viewing the website, or via the buttons on the Reset Password, Successful Reset Password, Contact Administrator, Register Account, Successful Account Creation, or Home Screen Pages. It's main purpose is to log into a user's account, but includes extra functionality. It includes 4 buttons and 2 text fields.

- Email: This text field will take email text input from the user.
- Password: This text field will take password text input from the user.
- Forgot Password: This button will navigate the user to a new page that will include instructions on how to recover their password.
- Login: This button will take the text inputs from the Username and Password text fields and attempt to use them to log into a user's account. Should login be successful, the user will be brought to the home screen. If it is unsuccessful an invalid username/password message will be displayed.
- Create Account: This button will navigate the user to the Account Creation Page.
- Contact: This button will navigate the user to the Contact Administrator Page.

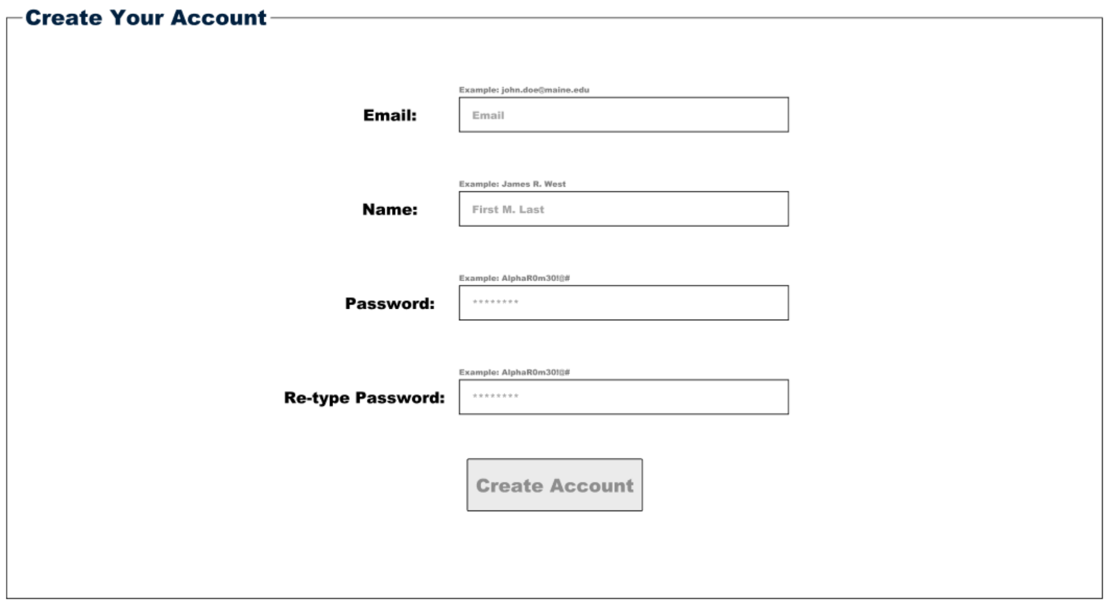
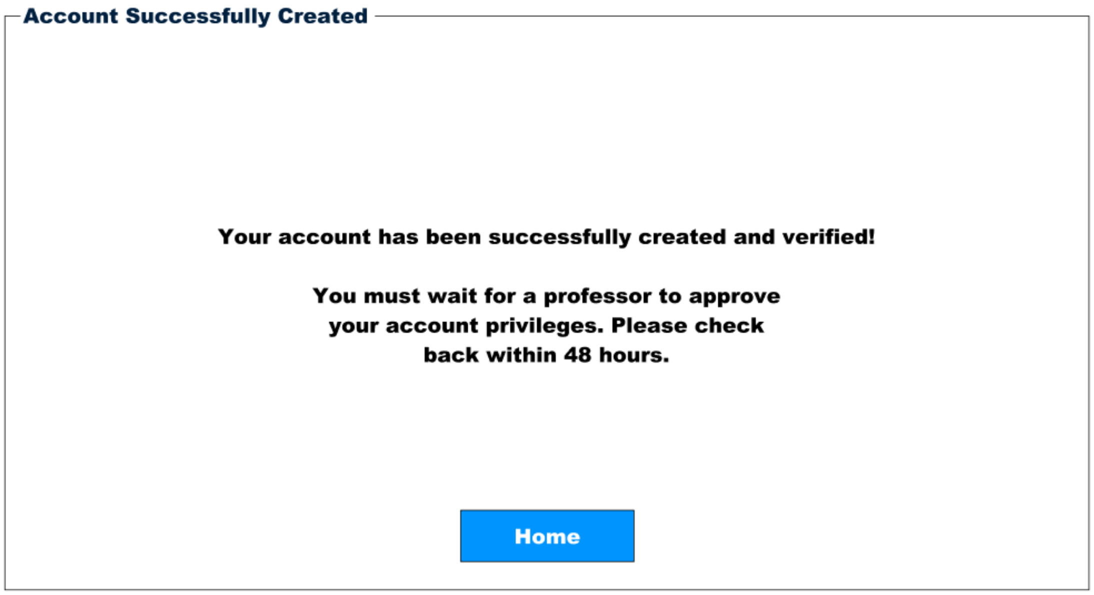


Figure 3.3: Create Account Page

The Create Account Page, Figure 3.3, is navigated to upon clicking the Create Account button on the Login Page. It's main purpose is to gather information required for a new user account and includes 4 text fields and 1 button.

- Email: This text field will take email text input from the user.
- Name: This text field will take name text input from the user.
- Password: This text field will take password text input from the user.
- Re-type Password: This text field will take retyped password text input from the user.
- Create Account: This button will take the information inputted in this page's text fields and check for validity. If information is not valid an error will be displayed to the user. If the input is valid and the two password inputs match each other, the information is saved and sent to an administrator for confirmation. The user is then sent to the Account Creation Success page.


Account Successfully Created

Your account has been successfully created and verified!

You must wait for a professor to approve your account privileges. Please check back within 48 hours.

Figure 3.4: Account Creation Success Page

This Account Creation Success Page, Figure 3.4, is navigated to by clicking on the Create Account button on the Create Account page with valid inputs. This page's primary function is to provide a message to the user about successful account creation and waiting for account approval. The page includes 1 button.

- Home: This button will navigate the user to the Login page.

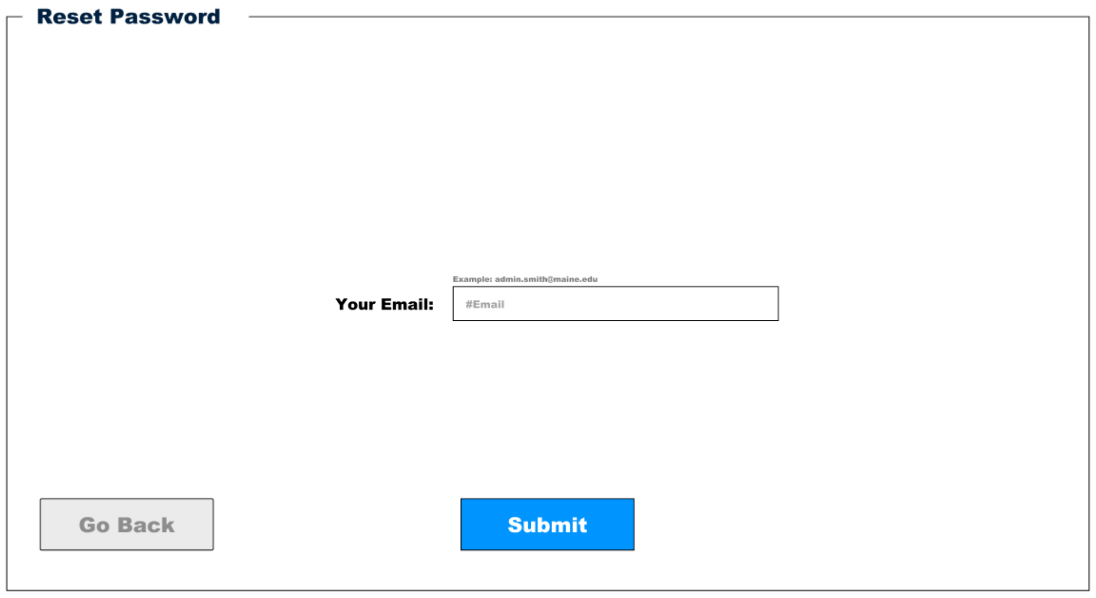
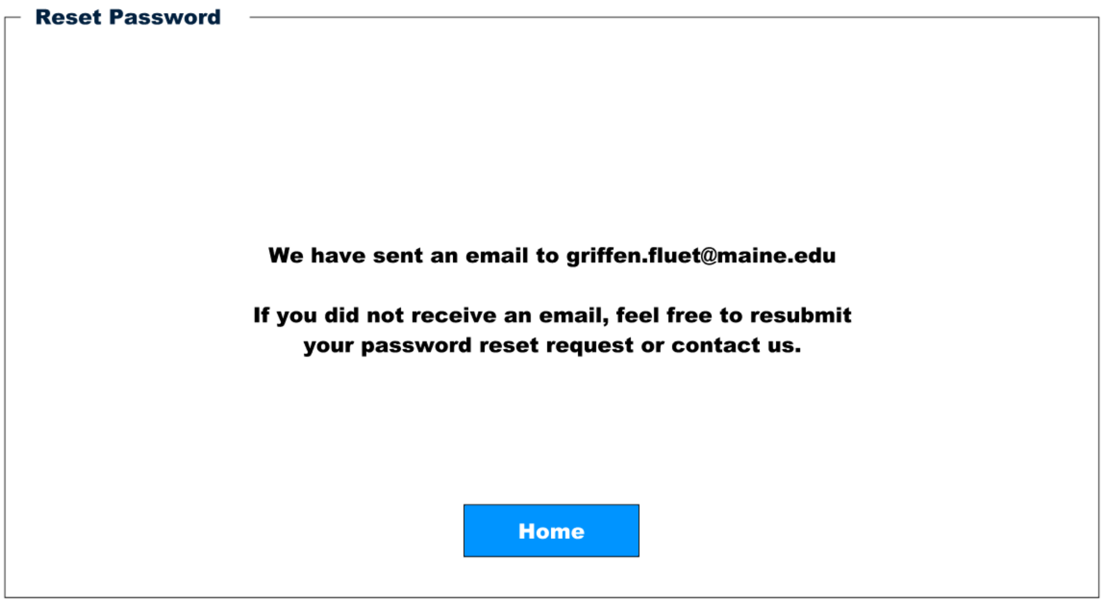


Figure 3.5: Reset Password Page

This Reset Password Page, Figure 3.5, is navigated to by clicking on the Reset Password button on the Login page. This page’s primary function is to reset a user's password after they have forgotten it. The page includes 1 text field and 2 buttons.

- Your Email: This text field takes email text input from the user.
- Submit: The submit button will take a user’s input from the Your Email text field and check if there is a user account with that email associated to it. If there is a match, that account’s password will be reset and the user will be navigated to the reset password confirmation page.
- Go Back: This button will navigate the user back to the Login page



We have sent an email to griffen.fluet@maine.edu

If you did not receive an email, feel free to resubmit your password reset request or contact us.

Figure 3.6: Reset Password Success Page

This Reset Password Success Page, Figure 3.6, is navigated to by clicking on the Submit button on the Reset Password page with valid input. This page's primary function is to provide a message to the user directing them to their email for further instruction. The page includes 1 button.

- Home: This button will navigate the user to the Login page.

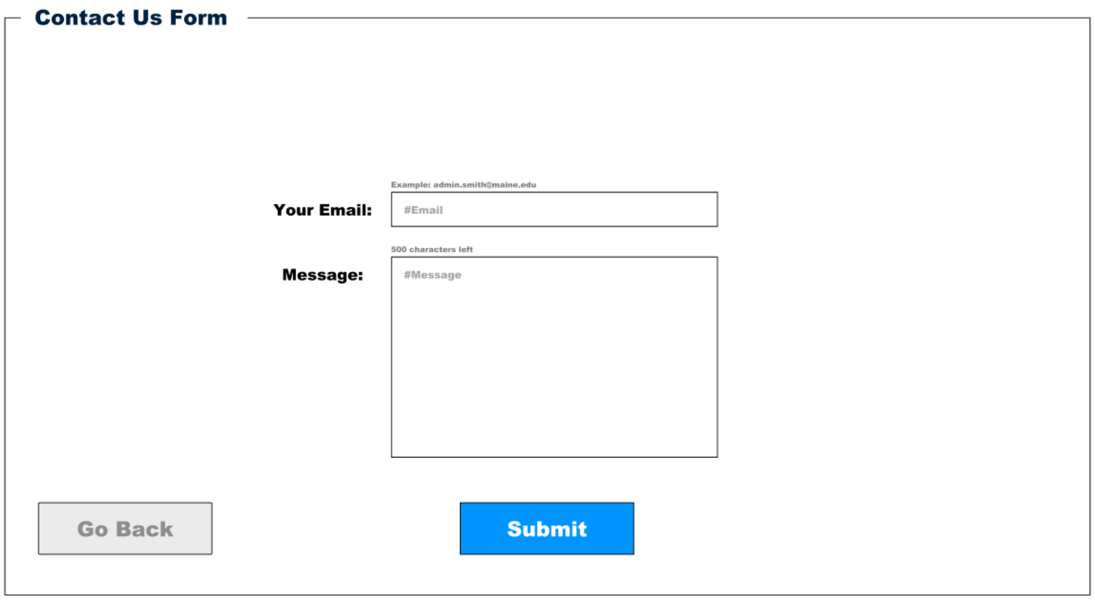


Figure 3.7: Contact Administrator Page

This Contact Administrator Page, Figure 3.7, is navigated to by clicking the Contact button on the Login page. This page's primary function is to allow the user to send a message to an administrator. The page includes 2 text fields and 2 buttons.

- Your Email: This text field takes email text input from the user.
- Message: This text field takes message input from the user.
- Submit: This button takes the text input from the two text boxes on this page and sends the information as a message to an Administrator.
- Go Back: This button will navigate the user back to the Login page.

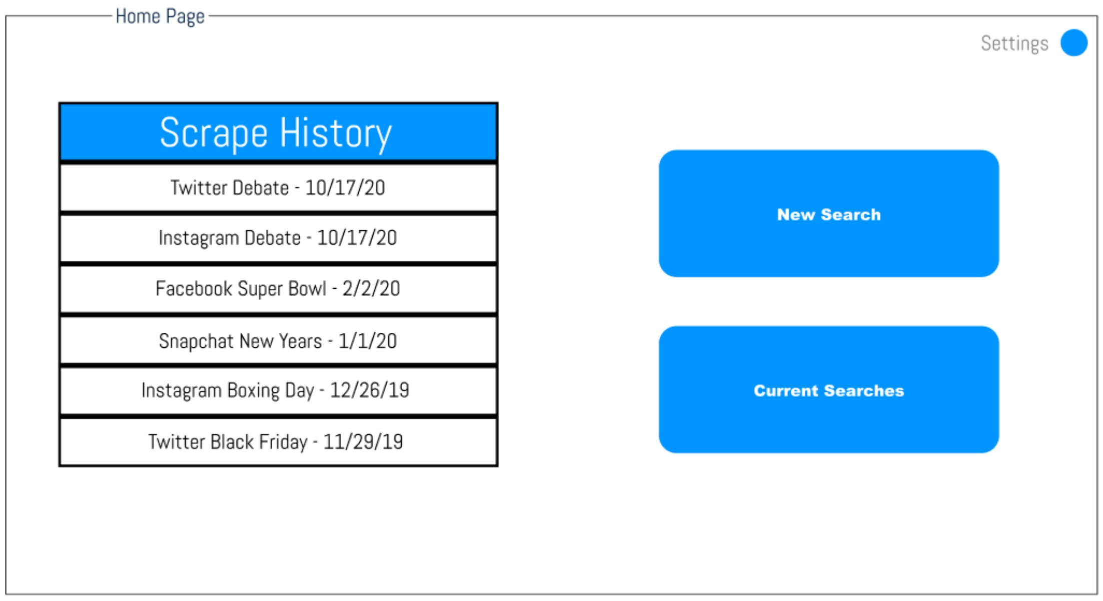
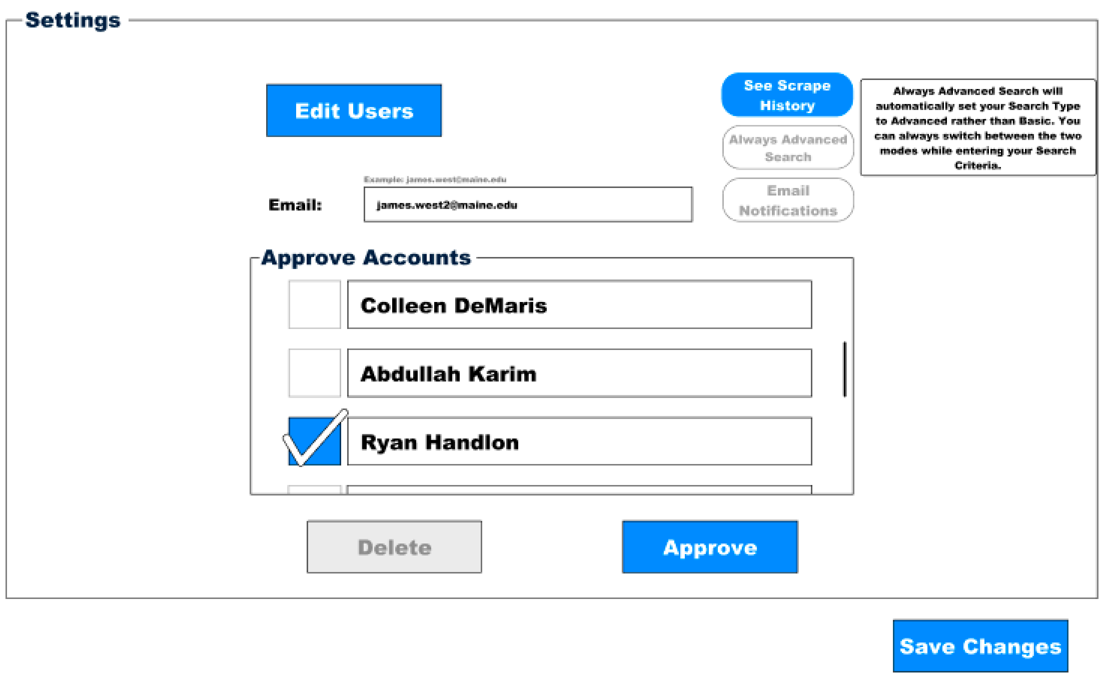

Home Page
Settings
Scrape History
Twitter Debate - 10/17/20
Instagram Debate - 10/17/20
Facebook Super Bowl - 2/2/20
Snapchat New Years - 1/1/20
Instagram Boxing Day - 12/26/19
Twitter Black Friday - 11/29/19
New Search
Current Searches

Figure 3.8:Home Page

The Home Page, Figure 3.8, is navigated to by clicking on the Login button on the Login page. It's primary function is to help navigate the user to the applications primary functions. The page has 3 buttons and one table with many buttons.

- New Search: This button will navigate the user to the New Search page.
- Current Searches: This button will only be visible if the user is logged into an Administrator account. When clicked it will navigate the user to the Current Searches page.
- Settings: This button will navigate the user to the settings page.
- Scrape History: This table will display a list of previous scrapes the user has done. Upon clicking one of these previous scrapes the user will be navigated to the new search page with that scrape's specific search results.



Always Advanced Search will automatically set your Search Type to Advanced rather than Basic. You can always switch between the two modes while entering your Search Criteria.

Example: james.west@maine.edu

Email: james.west2@maine.edu

	Colleen DeMaris
	Abdullah Karim
✓	Ryan Handlon

Figure 3.9: Admin Settings Page

The Admin Settings Page, Figure 3.9, is navigated to from the home page. This is the page that shows up for admin users. It allows the admin users to approve accounts, delete accounts, or to adjust personal email data and the location for the downloads. It also allows the user to set the search to an always advanced search. The user can also enable and disable email notifications.

- Edit Users: This button will bring the user to the Edit Users page.
- See Scrape History: Will enable/disable the scrape history on the home page.
- Always Advanced Search: Will enable/disable advanced search being the default search.
- Email Notifications: Will enable/disable email notifications for this user.
- Approve Accounts: This section will allow the admin user to approve or delete account requests.
- Save Changes: This button will save any changes made to the settings in a config file.

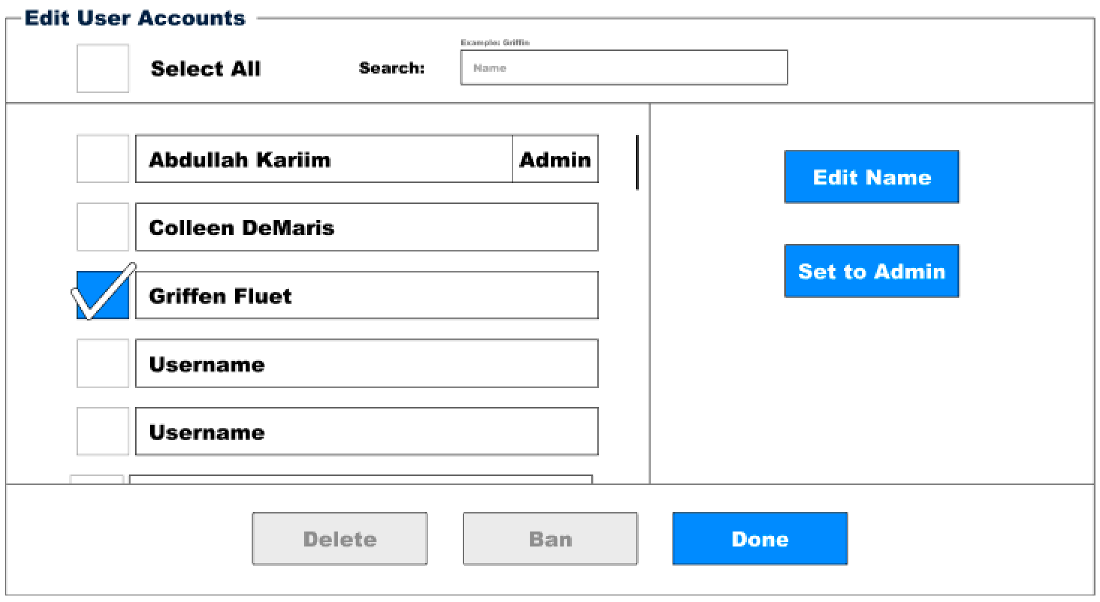
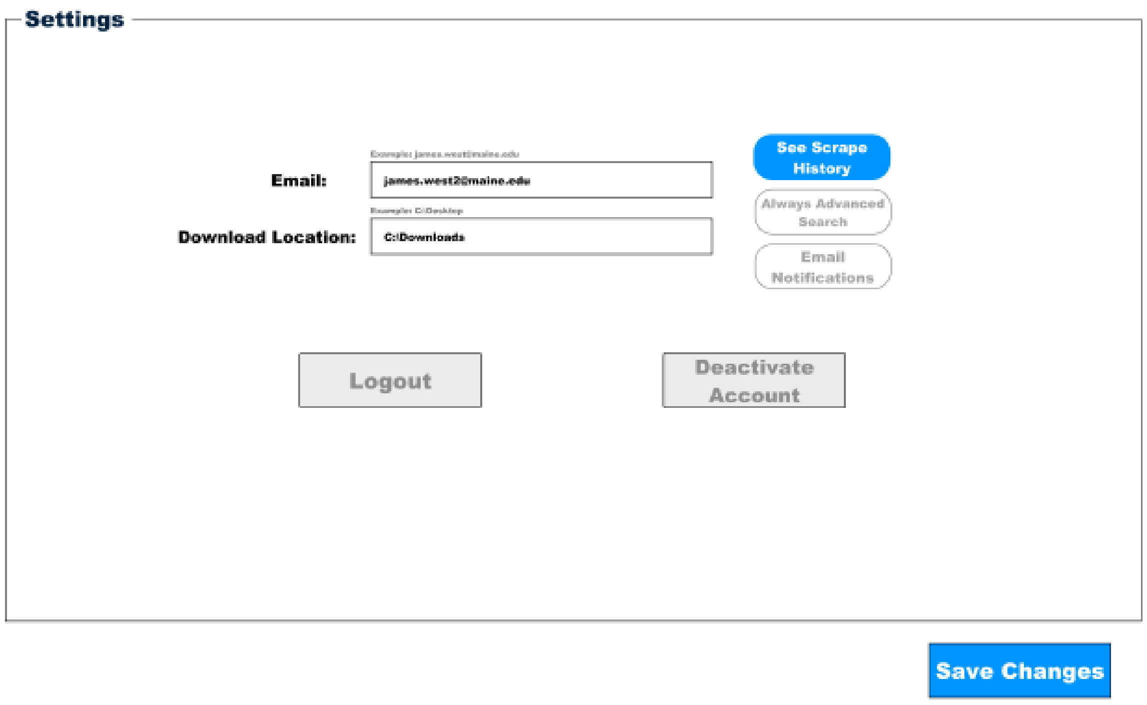


Figure 3.10: Edit Users Page

The Edit Users Page, Figure 3.10, is accessed via the Edit Accounts button on the Admin Settings page. It allows the user to delete, ban, or edit any other accounts, as well as see which accounts are admin and which are not.

- Delete: Deletes the selected account(s).
- Ban: Bans the selected account(s), but does not delete the information.
- Edit Name: Edits the name of the selected account(s).
- Set to Admin: Gives admin rights to the selected account(s).
- Done: Navigates to the previous page, the Admin Settings.



Example: james.west@maine.edu

Email: james.west2@maine.edu

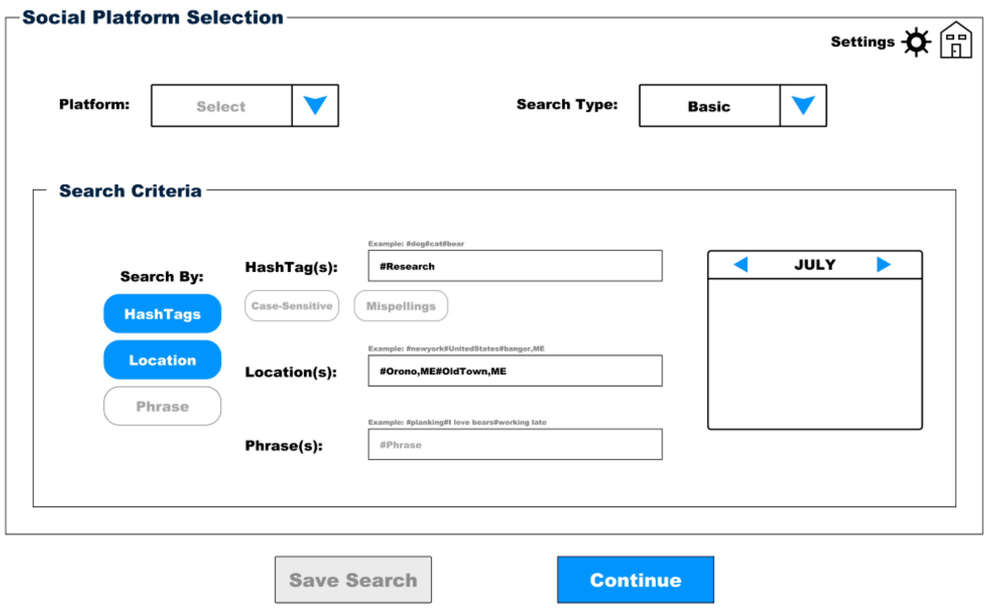
Example: C:\Desktop

Download Location: C:\Downloads

Figure 3.11: Researcher Settings Page

The Researcher Settings Page, Figure 3.11, is navigated to from the Home page. This is the page that shows up for non-admin users.

- Email: Allows the user to edit their email address (which is also used as their login).
- Download Location: Allows the user to change where the downloads go on their computer.
- See Scrape History: Will enable/disable the scrape history on the home page.
- Always Advanced Search: Will enable/disable advanced search being the default search.
- Email Notifications: Will enable/disable email notifications for this user.
- Logout: Logs out of the user's account.
- Deactivate Account: Deactivates and deletes the information of the user's account.
- Save Changes: Saves any changes made to the settings, and navigates to the previous page, the Home page.



Platform: Select
Search Type: Basic

HashTag(s): #Research
Example: #dog#cat#bear

Location(s): #Orono,ME#OldTown,ME
Example: #newyork#UnitedStates#bangor,ME

Phrase(s): #Phrase
Example: #planking#I love bears#working late

Figure 3.12: New Search Page

The New Search Page, Figure 3.12, is navigated to from the Home page. Its primary purpose is to start a new search, whether it be a basic or an advanced search. This page has two drop down menus, three different ways of searching (hashtags, location, and/or phrase), which can be searched for in any combination (as long as it includes at least one of the options), and boxes in order to fill out the search queries for the hashtags, location, or phrases.

- Platform: Allows the user to select which platform will be scraped.
- Search type: There are two options, basic or advanced. The advanced version of this will be shown on the next page.
- Hashtags button: Enables/disables search by hashtags.
- Location: Enables/disables search by location.
- Phrase: Enables/disables search by a phrase.
- Hashtags text field: Allows the user to enter hashtags to search for.
- Case Sensitive: Enables/disables case sensitivity with the search.
- Misspellings: Enables/disables letting the program look for common misspellings of the word(s).
- Location text field: Allows the user to enter locations to search for.
- Phrase text field: Allows the user to enter a phrase to search for.
- Save Search: This button saves the search to a database so that it can be searched for again or documented.
- Continue: Continues to the next page: the Searching page.
- Home: Navigates the user to the Home Page.

- Calendar: This calendar will allow the user to select a date range for the search specifications.

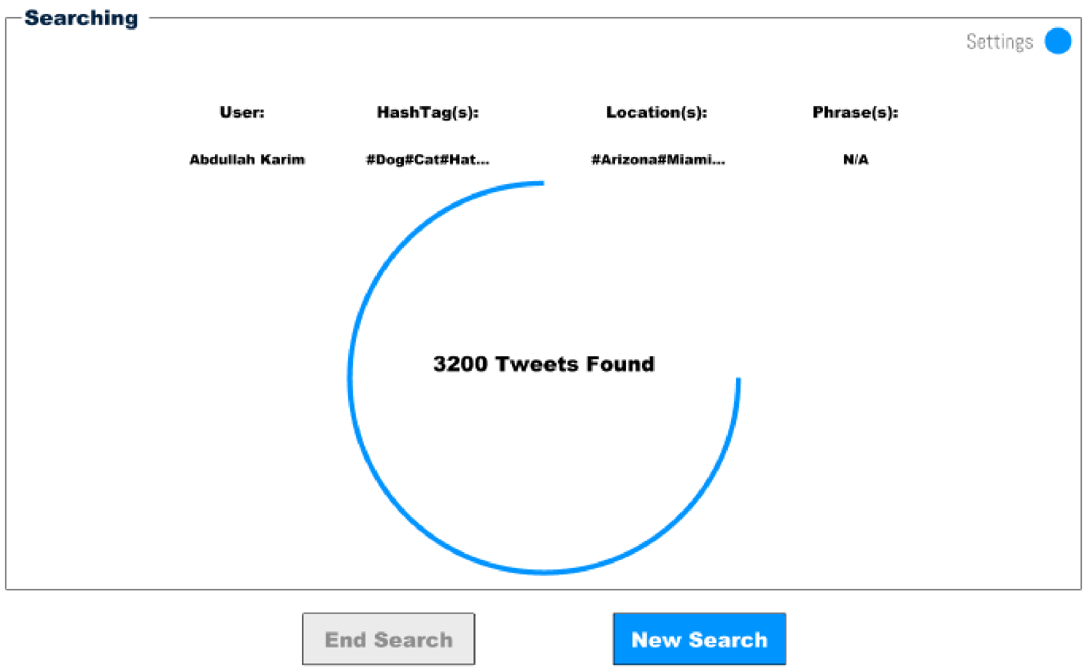
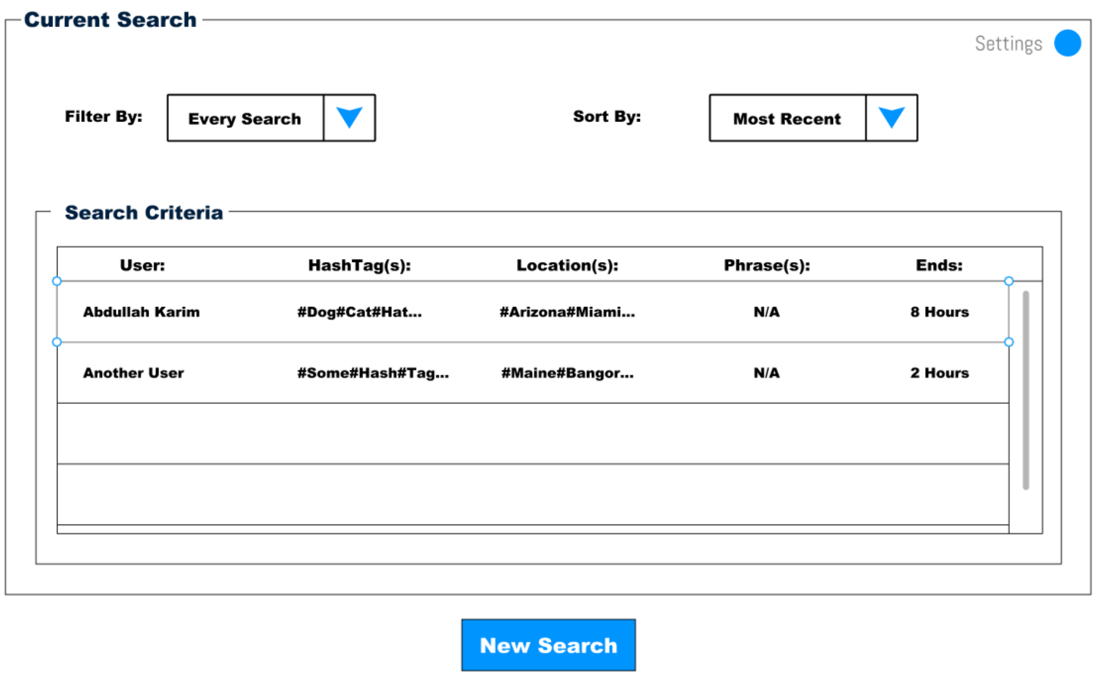


Figure 3.13: Searching Page

This is the Searching page, Figure 3.13. The primary purpose of this page is to show the scrape that is currently in progress, and show how many pieces of data (tweets, instagram posts, etc.) have been found.

- End Search: stops the search and finishes saving the data found.
- New Search: navigates to the Search page.
- Settings: navigates to the settings page, depending on whether the user account is an admin or a basic account.



User:	HashTag(s):	Location(s):	Phrase(s):	Ends:
Abdullah Karim	#Dog#Cat#Hat...	#Arizona#Miami...	N/A	8 Hours
Another User	#Some#Hash#Tag...	#Maine#Bangor...	N/A	2 Hours

Figure 3.14: Current Search Page

The Current Search Page, Figure 3.14, is navigated to by clicking on the Current Searches button on the on the Home page. It's primary function is to display all searches being run on the platform at the current time. The Page has 2 buttons, 2 drop down boxes, and one table.

- New Search: This button will navigate the user to the New Search page.
- Settings: This button will navigate the user to the settings page.
- Filter By: This drop down box will allow the user to filter the current searches by Every Search, a Specific User, or Social Media Platform
- Sort By: This drop down box will allow the user to sort the current searches by Most Recent, Least Recent, Number of Posts Scraped, and Estimated Time to Completion.

4. Data Validation

This section shows the data a user will be able to enter into our product. It includes a unique identifier, corresponding GUI screens (see above sections), the formatting of data, and the limits for each data input.

Table 4.1: Data Input Validation Specifications

Unique ID	Data Type	Data Format	Limits	GUI Screen (s)
Email	String	[Email]@[domain].[extension]. Cannot be empty.	100 chars	1. Reset Password 2. Settings 3. Create Your Account 4. Contact Us Form 5. Scraper Log In
Download Location	String	[Drive]\[filepath]\...\[destination]. Cannot be empty.	500 chars	1. Settings
Search Users	String	Any acceptable string that follows a string regex checking if it's empty.	50 chars	1. Edit User Accounts
Contact Message	String	Any acceptable string that follows a string regex checking if it's empty.	500 chars	1. Contact Us Form
Password	String	Any acceptable string that follows a string regex checking if it's empty. Created passwords must have a capital letter, a lowercase letter, a number, and a special symbol. Passwords must also be at least 8 characters long.	40 chars	1. Scraper Log In 2. Create Your Account
Name	String	[First Name] [Middle Initial]. [Last Name]. Must follow this format.	60 chars	1. Create Your Account
HashTag(s) Field	String	Each hashtag follows the format: #[string]. Multiple hashtags can be input as long as	60 chars	1. Social Platform Selection

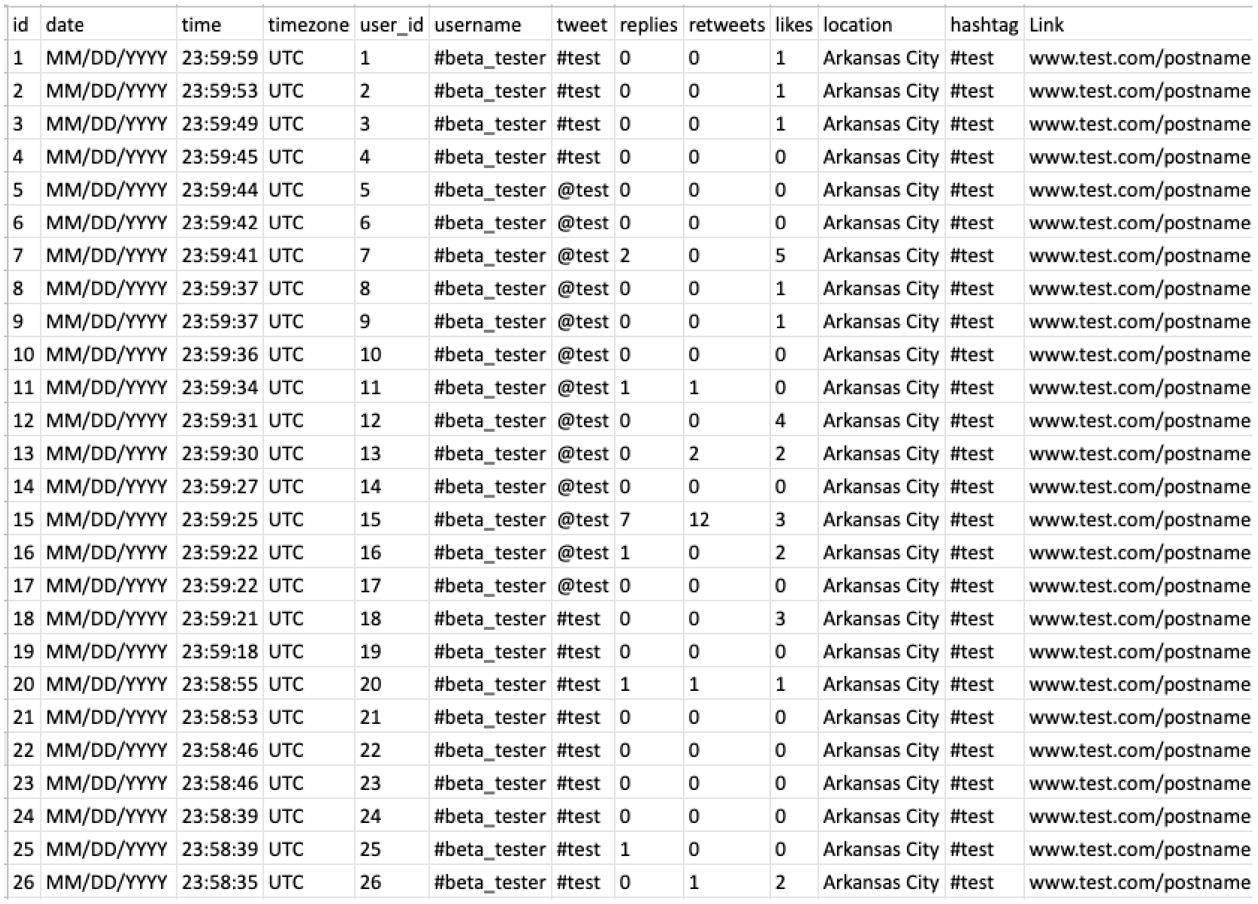
		they are comma-separated.		
Location(s) Field	String	Each Location can be entered as a string, with multiple locations being designated as comma-separated values.	100 chars	1. Social Platform Selection
Phrases(s) Field	String	Each Phrase can be entered as a string, with multiple phrases being designated as comma-separated values.	280 chars	1. Social Platform Selection
Run For Field	integer	Must be between 1 and the maximum signed integer limits.	10 chars	1. Social Platform Selection - Advanced Search

5. Report Formats

This section covers the hard copies that our users will get when downloading scraped information.

Our product generates a downloadable .zip file. This .zip contains a folder of all images scraped as well as a .csv file for text data. This .zip has its name made up from the keywords of the search. The name of each picture file corresponds to the unique ID of each picture in a .csv file discussed below.

A separate .csv file is located at the same level as the image folder directory. It contains all text data from a scrape. Figures of the formatting for both the .csv file and the picture file system are found below:



id	date	time	timezone	user_id	username	tweet	replies	retweets	likes	location	hashtag	Link
1	MM/DD/YYYY	23:59:59	UTC	1	#beta_tester	#test	0	0	1	Arkansas City	#test	www.test.com/postname
2	MM/DD/YYYY	23:59:53	UTC	2	#beta_tester	#test	0	0	1	Arkansas City	#test	www.test.com/postname
3	MM/DD/YYYY	23:59:49	UTC	3	#beta_tester	#test	0	0	1	Arkansas City	#test	www.test.com/postname
4	MM/DD/YYYY	23:59:45	UTC	4	#beta_tester	#test	0	0	0	Arkansas City	#test	www.test.com/postname
5	MM/DD/YYYY	23:59:44	UTC	5	#beta_tester	@test	0	0	0	Arkansas City	#test	www.test.com/postname
6	MM/DD/YYYY	23:59:42	UTC	6	#beta_tester	@test	0	0	0	Arkansas City	#test	www.test.com/postname
7	MM/DD/YYYY	23:59:41	UTC	7	#beta_tester	@test	2	0	5	Arkansas City	#test	www.test.com/postname
8	MM/DD/YYYY	23:59:37	UTC	8	#beta_tester	@test	0	0	1	Arkansas City	#test	www.test.com/postname
9	MM/DD/YYYY	23:59:37	UTC	9	#beta_tester	@test	0	0	1	Arkansas City	#test	www.test.com/postname
10	MM/DD/YYYY	23:59:36	UTC	10	#beta_tester	@test	0	0	0	Arkansas City	#test	www.test.com/postname
11	MM/DD/YYYY	23:59:34	UTC	11	#beta_tester	@test	1	1	0	Arkansas City	#test	www.test.com/postname
12	MM/DD/YYYY	23:59:31	UTC	12	#beta_tester	@test	0	0	4	Arkansas City	#test	www.test.com/postname
13	MM/DD/YYYY	23:59:30	UTC	13	#beta_tester	@test	0	2	2	Arkansas City	#test	www.test.com/postname
14	MM/DD/YYYY	23:59:27	UTC	14	#beta_tester	@test	0	0	0	Arkansas City	#test	www.test.com/postname
15	MM/DD/YYYY	23:59:25	UTC	15	#beta_tester	@test	7	12	3	Arkansas City	#test	www.test.com/postname
16	MM/DD/YYYY	23:59:22	UTC	16	#beta_tester	@test	1	0	2	Arkansas City	#test	www.test.com/postname
17	MM/DD/YYYY	23:59:22	UTC	17	#beta_tester	@test	0	0	0	Arkansas City	#test	www.test.com/postname
18	MM/DD/YYYY	23:59:21	UTC	18	#beta_tester	#test	0	0	3	Arkansas City	#test	www.test.com/postname
19	MM/DD/YYYY	23:59:18	UTC	19	#beta_tester	#test	0	0	0	Arkansas City	#test	www.test.com/postname
20	MM/DD/YYYY	23:58:55	UTC	20	#beta_tester	#test	1	1	1	Arkansas City	#test	www.test.com/postname
21	MM/DD/YYYY	23:58:53	UTC	21	#beta_tester	#test	0	0	0	Arkansas City	#test	www.test.com/postname
22	MM/DD/YYYY	23:58:46	UTC	22	#beta_tester	#test	0	0	0	Arkansas City	#test	www.test.com/postname
23	MM/DD/YYYY	23:58:46	UTC	23	#beta_tester	#test	0	0	0	Arkansas City	#test	www.test.com/postname
24	MM/DD/YYYY	23:58:39	UTC	24	#beta_tester	#test	0	0	0	Arkansas City	#test	www.test.com/postname
25	MM/DD/YYYY	23:58:39	UTC	25	#beta_tester	#test	1	0	0	Arkansas City	#test	www.test.com/postname
26	MM/DD/YYYY	23:58:35	UTC	26	#beta_tester	#test	0	1	2	Arkansas City	#test	www.test.com/postname

Figure 5.1: .csv File Formatting

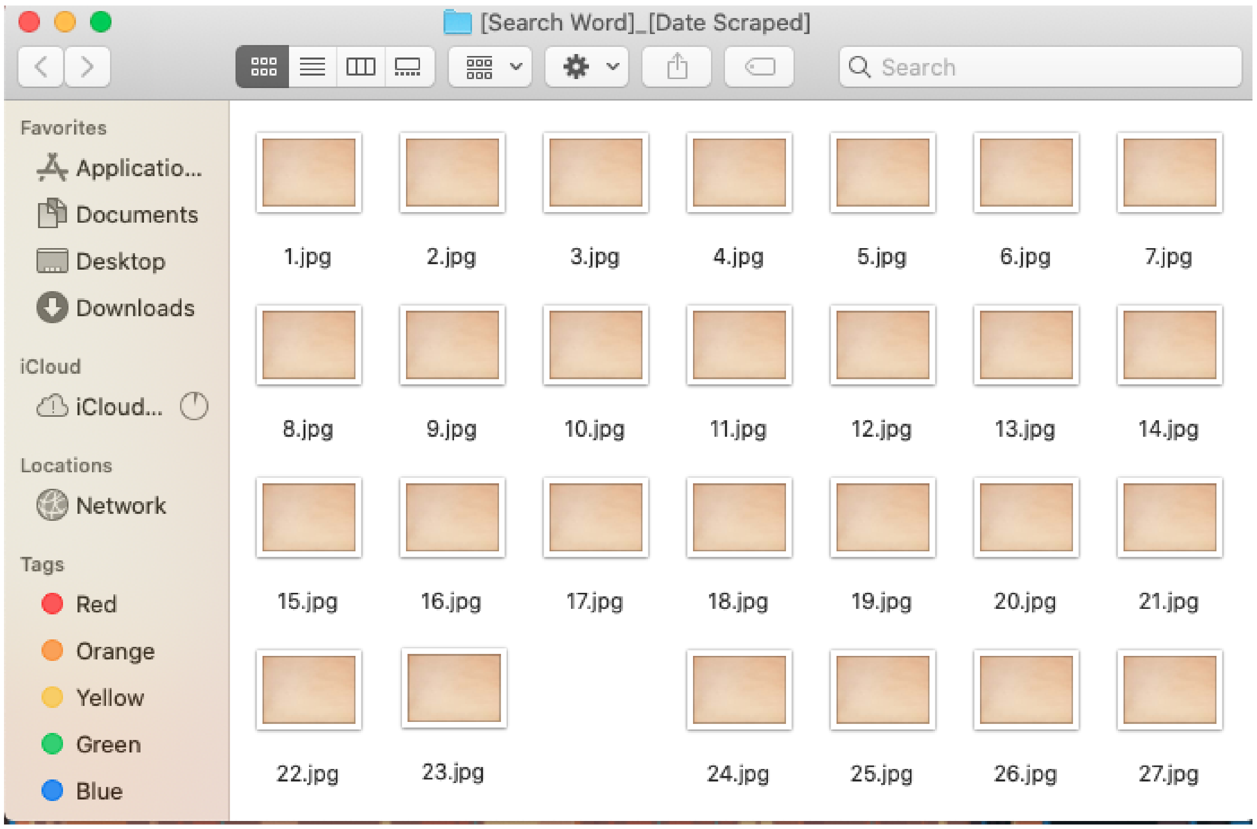


Figure 5.2: Picture File System Formatting

Appendix A – Agreement Between Customer and Contractor

The client (Doctor Rosenbaum) and Blackbear Analytics agree that the system's design meets all requirements prescribed in the SRS document and will be implemented in good faith. All designs will be implemented as close to the specifications in the user interface design document as possible, with margins for error being slim or none. In the event that implementation must diverge from the design, the development team will notify the client as soon as possible and work together with the client until a suitable agreement is reached. If the client would like to add designs/requirements for the system's user interface design after signing this document, Blackbear Analytics bears no responsibility for the failure to implement new designs, but will attempt to add them in good faith. Should any new designs be added, the deliverables schedule in section 2 and 3 of the SRS document will be updated immediately to reflect changes, and the designs/requirements discussed will be reflected in this document by the next deliverable deadline and changes will be finalized with the client.

Name	**Signature**	**Date**
Customer:		
Comments:		
Team:		
Ryan M. Handlon	[illegible]	November 24th, 2020
Comments:		
Abdullah I. Karim	[illegible]	November 24th, 2020
Comments:		
Griffin L. Fluet	Griffin L. Fluet	November 24th, 2020
Comments:		
Colleen DeMaris	Colleen DeMaris	November 24th, 2020
Comments:		
James West	James West	November 24th, 2020

Comments:

Appendix B – Team Review Sign-off

By signing your name below, you acknowledge that you are a member of Blackbear Analytics and have read the document with an in-depth comprehension of the scribed materials. You agree to complete all designs stated on this document as is, in good faith. Should any new designs come up, you agree to assist in re-drafting this document for approval as described in Appendix A. You agree that it is not required to complete designs added after this version is released, but understand that all designs, new or old, must be implemented in good faith.

Ryan M. Handlon	[illegible]	November 24th, 2020

Comments:

Abdullah I. Karim	[illegible]	November 24th, 2020

Comments:

Griffin L. Fluet	Griffin L. Fluet	November 24th, 2020

Comments:

Colleen DeMaris	Colleen DeMaris	November 24th, 2020

Comments:

James West	James West	November 24th, 2020

Comments:

Appendix C – Document Contributions

Ryan, Abdullah, Griffin, and Colleen contributed 16.25% total to this document. James contributed 35% because he headed nearly all the design for the mockup on figma. Ryan Handlon created half of the User Interface Walkthrough (Section 3). Abdullah Karim helped in the creation of Data Validation (Section 4) and Report Formats (Section 5). Griffin Fluet contributed massively to the User Interface Standards (Section 2). Colleen DeMaris did half of the User Interface Walkthrough (Section 3). James West created the mockup on figma and helped with the User Interface Standards (Section 2). The Appendices were appended and modified from the SRS document and Appendix C have been read and reviewed by the whole team.

Appendix D - CIR

Product: S.S. Media

Client: Doctor Judith E. Rosenbaum

Code Inspection Report

BLACK BEAR
ANALYTICS

Black Bear Analytics

Abdullah Karim | Colleen DeMaris | Griffin Fluet | James West | Ryan Handlon

March 17th, 2020

Revision History

Version Number	Release Date	Description
Version 1.0	03/17/2020	Original Release

S. S. Media

Code Inspection Report

Table of Contents

S. S. Media Abstract

Social media has grown to be an extremely large part of society in recent years. Because of this, a need to gather data from it has also arisen. Black Bear Analytics is creating a tool that allows researchers to scrape public data from posts on Twitter and Instagram. This tool was requested by Doctor Judith Rosenbaum of UMaine's Department of Communication and Journalism. The S.S. Media will have an easy-to-navigate interface that allows users to switch between scraping public posts on Instagram and Twitter. The *New Search* page gives the user an option of either an advanced search or a basic search. The basic option searches by specified hashtags, locations, or phrases, and an acceptable start and end date to check with each post scraped, while an advanced search has the same general functions as the basic search, but adds the ability to search for more than one topic (hashtag, location, phrase), as well as run the search once every specified amount of time. The tool will store a number of scrape requests, so that users can either inspect a previous scrape request or request it again. Once the scraping is completed, the data will be downloaded to the user's computer in the form of a .csv file, and images will be stored in a corresponding folder. The outcome of this project will hopefully be that researchers like Doctor Rosenbaum can have a tool that will enable them to easily search for data and trends on social media sites.

1. **Introduction**

The purpose of this section is to introduce the reader to both this document and the Code Inspection Review, detailing introductory information that will be useful for the reader to know.

1.1 Purpose of This Document

The purpose of this document is to report the conclusion of an egoless code review conducted with defined coding conventions. Black Bear Analytics met several times to refine the coding conventions in this document to present to the reader in conjunction with defects identified during code review.

1.2 References

Black Bear Analytics SRS Document

Black Bear Analytics SDD Document

1.3 Coding and Commenting Conventions

The coding and commenting conventions defined in Appendix A seek to reduce readability errors within the code and enhance the future code review process. Black Bear Analytics took initiative in creating the coding conventions based on prior experience and models seen as the industry standard. Industry standard conventions can be found in the following references:

Admin. (2018, March 28). *Coding Standards and Best Practices*. Aversan. http://www.aversan.com/coding-standards-and-best-practices-2/.

Coding standards and guidelines. (2019, July 2). http://www.geeksforgeeks.org/coding-standards-and-guidelines/.

Black Bear Analytics did not modify any of the coding conventions listed in reference. The conventions are either created by Black Bear Analytics or adopted from the listed references. Please see Appendix A for coding and commenting conventions used in our project.

1.4 Defect Checklist

Below is an overview of the defects Black Bear Analytics uncovered during code review.

Table 1.1 - Defect Checklist by Category

Num.	Name	Module No.	Convention	Logic Flaw	Security	Comments
1.	Login User/Pass Scrub	1			■	
2.	Chrome Data Breach Popup	7			■	

3.	Backend User Credentials	4-21		■		
4.	Emoji Scrape Errors	1		■		
5.	Archived Write Data Loss	1		■		
6.	Category 2 Redundancy	2		■		
7.	Browser Waiting	2	■			
8.	Sleeps In Code	2,3		■		
9.	Comment Conventions	All				■
10.	Code Conventions	All	■			
11.	Fetch requests deformed	7-21		■		
12.	Login Auth Fails	7		■	■	
13.	Database displays	5			■	
14.	Scrape Twitter not correctly parsing input	6		■		
15.	Scraping more tweets than requested	1		■		
16.	Checking for user already in database	6		■		
17.	Deployment Pipeline Breakage	23		■		

2. **Code Inspection Process**

The purpose of this section is to give an overview of the Code Inspection Process that Black Bear Analytics took in it's most recent Code Review, as well as their general impressions of it.

2.1 Description

We started the review with our backend system and then switched back and forth with the frontend discussing the coding conventions to follow versus what was done. This gave each of the code developers ideas as to what refactoring had to be done. We listed those coding conventions here in Appendix A and continued to look for logical errors in the Instagram/Twitter Scraping. The UI was tackled after and we uncovered any defects a user might see. We then looked at the restAPI for defects in communication with our database and front-end to back-end systems.

In each review module, we began with naming conventions and then commenting conventions. Then we moved on to walking through the sections of the code, discussing what each piece does and what needs to be done on the backend. This process was repeated during the whole review to uncover any defects. We also took a look at our deployment and identified its degradation in the review.

2.2 Impressions of the Process

The code inspection process was very useful in bringing the team together, getting to walk through each piece of the project was insightful. Each team member was able to add valuable input towards finding bugs, creating standards to follow, and what pieces required refactoring. Now that we know how a code inspection works, we would have created commenting and naming conventions from the start.

The worst parts of our program are the pieces we still have not worked on yet as there is still a lot of work to be finished. We also do not have the Login Page verification working, a critical component for accessing the application. Our main issues revolve around code not being implemented yet.

2.3 Inspection Meetings

Below is a list of all inspection meetings Black Bear Analytics committed to for the review process:

03.11.2021: Met for code inspection.
Location: virtual, on discord
Time Started: 5:30pm
Time Ended: 8:30pm
Participants: Abdul, Colleen, Griffin, James, Ryan.

    Abdul: scribe for commenting conventions
    Colleen: scribe for meeting details
    Griffin: reader for his code, inspector for other code

James: inspector

Ryan: reader for his code, inspector for other code

03.15.2021: Met for code inspection. Location: virtual, on discord

Time started: 5pm

Time ended: 7pm

Participants: Abdul, Colleen, Griffin, James, Ryan

Abdul: reader for his code, inspector for other code

Colleen: scribe, inspector

Griffin: inspector

James: reader for his/colleen's code, inspector for other code

Ryan: inspector

3. **Modules Inspected**

The purpose of this section is to provide the reader with a brief understanding of all modules in the current software project that were inspected, and any relevant information that should accompany them.

The following modules appear in the SDD: URLPostExtractor, KeywordURLExtractor, User Interface, DOM Queue, and Page Processor. URLPostExtractor and KeywordURLExtractor differ from in the SDD, and are described in their modules. The User Interface is different from the SDD because we broke down the modules more. Instead of the User Interface, we have individual modules for each page that will be added into the S.S. Media.

Module 1: TweetExtractor.py

- Incomplete
- Projected Completion Date: 03.25.21
- Need to finish: Find a less resource intensive method of collecting tweets.
- Description: This module creates the query to send to Twitter's API based on input of hashtags, locations, and phrases. The returned tweets from the API call will have the data points of interest scraped and if a form of media exists in the tweet, it will be downloaded. Each tweet written to the CSV file receives an ID that will match the name of the media associated with it. In the case where there are more than one media in a tweet, there is a file for each one and have an incrementing value concatenated to the end of the file name. The result of this module will be zipped archive with the scraped data points and media.
- Difference from SDD: This module differs from the SDD since it is acting as both the KeywordURLExtractor and PostExtractor. When designing the SDD it was not determined if Twitter's API was going to be used, so the document was written in the context of scraping based off of HTML. This way of scraping Twitter makes it much easier to scrape more tweets than HTML based scraping. Since we are instead querying Twitter for the tweets that have a match based on hashtags, location, phrase, and even date ranges.

Module 2: InstagramKeywordURLExtractor.py

- Incomplete
- Projected Completion Date: 03.25.21
- Need to finish: This module currently works but needs to be refactored. The current scraping process needs to be modified for better handling of when the bottom of Instagram's explore page is reached, all sleep function calls removed, browser.implicitly_wait() needs to be rethought to account for slow internet, and the link extraction should be writing to the URLFrontier.txt file during scraping rather than after.
- Description: This module opens up a headless chrome browser navigated by Selenium. It logs into Instagram, and then visits the Instagram explore page for a specific hashtag or location. From there it scrolls down the explore page collecting links to posts and populating them into a text document.

Module 3: PostExtractor.py

- Incomplete
- Projected Completion Date: 04.10.21
- Need to finish: Using multiple accounts for each user for when an account gets blocked from the explore page. Also need to implement error handling when an Instagram post is deleted and we try scraping it.
- Description: This module grabs the links from the URL_Frontier and scrapes the wanted data points from an Instagram post and writes them to a CSV file. The image or video from that post is downloaded to a media directory. Each post written to the CSV file receives an ID that will match the name of the media associated with it. The result of this module is a zipped archive with the CSV file and media directory.

Module 4: ScrapeManager.py

- Incomplete
- Projected Completion Date: 04.10.2021
- Need to finish: Work on this module has not been started yet.
- Description: System to manage each scrape within a thread to allow for multiple scrapes to be going at the same time. Scrapes will start and end here and have an id based off of the user that started the scape. This will allow us to pass back information like the number of tweets scraped during a running scrape.
- Difference from SDD: This is a new module that we found a need for since we overlooked the need for managing the scrapes when a user starts one.

Module 5: User Database Schema

- Incomplete
- Projected Completion Date: 04.20.2021
- Need to Finish: Adding columns, foreign keys, and tables to reflect recent scrape history.
- Description: This is the database used to store user information. It fits into the SDD at the login section where we are storing user information. Haven't finished adding necessary columns to store all user information. We need to include a table or another column storing the users recent searches. There may be other information that needs to be stored that we haven't come across yet and this is an unknown risk to storing live user information.

Module 6: Endpoints

- Incomplete
- Projected Completion Date: 4.15.2021
- Need to Finish: Adding endpoints that the front-end can use to get information and the backend can use
- Description: This is the connection between the frontend and the backend. This fits into our SDD where any processing makes a backend or frontend call. The API we've created is missing some endpoints that are required for accessing the database. We lack the functionality to edit a user's entry, which will be necessary when saving recent searches to

the user database which also does not have an endpoint. There are also many endpoints that will need to be created as more of the project is implemented such as an instagram scrape endpoint and other endpoints that facilitate the communication between front-end and backend.

Module 7: Login Page

- Incomplete
- Projected Completion Date: 04.10.2021
- Need to Finish: The connection to the restAPI we have is configured but does not send the user to the HomePage based on verification or give a message barring them from entry because of invalid credentials.
- Description: This is the landing page for all users to start at. It fits into the start of our design from the SDD because users need a place to input credentials for a secure connection. All of the styling is done for this page along with the input sections that the user can enter information into.

Module 8: Search Criteria

- Incomplete
- Projected completion date: 04.10.2021
- Need to finish: Creating CSS and ReactJS files for both the advanced and the basic options. Need to connect to the database in order to store the search criteria. Need to connect to the scraper back end to initiate the search with the filled out information.
- Description: This page displays the options to fill out a scrape search. The user can choose between the different platforms, Twitter and Instagram, in a drop down menu. The user can also choose whether they would like a basic or advanced search. The advanced search allows the user to search using time intervals, from one date to the next every given amount of time. The advanced search also includes everything on the

Module 9: Edit Users Page

- Incomplete
- Projected completion date: 04.10.2021
- Need to finish: Creating the CSS and ReactJS files. Need to then connect to the database so that it can display the list of all accounts.
- Description: This page is the Edit Users page for admin users of the S.S. Media. It is navigated to from the Admin Settings page. This page displays a list of accounts with information on whether they are an admin or not. The admin looking at the account list can change who is an admin and who is not. The admin user can also ban accounts from this page.

Module 10: Home Page

- Incomplete
- Projected Completion Date: 03.18.2021

- Need to Finish: Adding buttons and navigation
- Description: This is the page that the user can go to for accessing their scrape history, conducting a new search, or looking at current searches. This is the hub where users can access nearly all other pages in our website and currently only has one button to start a new search on it.

Module 11: Searching Page
- Incomplete
- Projected completion date: 04.10.2021
- Need to finish: Creating CSS and ReactJS files. Need to connect the page to the back end extractor, to display the number of posts found. Also need to be able to display the scraping criteria above the number of posts found.
- Description: This page is displayed after a scrape is initiated. It displays the scrape criteria that the user put in, as well as a loading circle with the number of posts found and scraped. The user has the options to either end the search, start a new search, go home, or enter the settings.

Module 12: Admin Settings Page
- Incomplete
- Projected completion date: 04.10.2021
- Need to finish: The CSS and ReactJS have been created, need to connect the page with the database to display the user accounts that are pending and must be approved. Must also create functionality for each button. Need to hook up the page to a .config file in order to store saved settings. Need to be able to navigate to the page.
- Description: This page is the settings page for admin users of the S.S. Media. The page gives users the ability to view any contact requests that were sent in, edit the users that are allowed access and what type of access they are allowed, deactivate the account, or change the scrape history, advanced search, and email notification toggles. This page displays accounts that must be approved so that the admin user can approve them or delete their requests. Users can also logout from this page.

Module 13: Settings Page
- Incomplete
- Projected completion date: 04.10.2021
- Need to finish: The CSS and ReactJS have been created, need to connect the page to a .config file in order to store saved settings. Need to be able to navigate to the page.
- Description: This page is the settings page for non-admin users of the S.S. Media. The page gives users the ability to toggle their scrape history visuals on/off, the advanced search on/off as default, and email notifications on/off. The user can deactivate their account from this page, or log out.

Module 14: Account Requested Page

- Incomplete
- Projected completion date: 04.10.2021
- Need to finish: Creating CSS and ReactJS files. Need to be able to navigate to the page.
- Description: This page is the page that is displayed when someone first makes a new account. Once the account information is filled out, this page is displayed and the user can only go back to the login page until their account is approved by an admin.

Module 15: Register Account Page

- Incomplete
- Projected completion date: 04.10.2021
- Need to finish: Creating the CSS and ReactJS files. Need to be able to navigate to the page. Need to connect it to the database in order to store the account information.
- Description: This is the page that displays when a user clicks the Create button on the login page. It takes in an email, name and password, and sends it to the database.

Module 16: Current Searches Page

- Incomplete
- Projected completion date: 04.10.2021
- Need to finish: CSS styling and ReactJS files, Linking using React-Router.
- Description: This is the page that displays the searches that are currently being run. The user can choose filters from a drop down menu as to what previous searches are displayed.

Module 18: Reset Password Notification Page

- Incomplete
- Projected completion date: 04.20.2021
- Need to finish: CSS styling and ReactJS files, link to home login page.
- Description: This page displays a message that an email has been sent to reset the user's password. It has one button, Home, that brings the user to the login page.

Module 19: Reset Password Message Page

- Incomplete
- Projected completion date: 04.20.2021
- Need to finish: CSS styling and ReactJS files, link to function that sends email to the user to reset the password. Link to Reset Password Notification Page to display message.
- Description: this page has one input that takes the user's email address. It has a Go Back button that directs back to the login page. It has a submit button that will send the email address to the function that will send an email to that address. The submit button will also redirect the user to the notification page.

Module 20: Register Account Success Page

- Incomplete
- Projected completion date: 04.20.2021

- Need to finish: CSS styling and ReactJS files.
- Description: This page is displayed to inform the user of their new account when the user has entered valid input on the Register Account Page, and clicks the Create Account button.

Module 21: View Contact Requests Page

- Incomplete
- Projected completion date: 04.20.2021
- Need to finish: CSS styling and ReactJS files.
- Description: This page is accessed from an Admin user account and displays the contact requests made by non-users that filled out the Contact Us Page's form.

Module 23: Full-Stack Deployment Pipeline

- Incomplete
- Projected Completion Date: 03.20.2021
- Need to Finish: Adding deployment details for Back-end product
- Description: This is the pipeline we are using to deploy our software to AWS as seen in the SDD.

4. **Defects**

The purpose of this section is to list all defects found during formal code review and coding convention definition and provide information about them. They are numbered in accordance with the defect checklist (Table 1.1):

1. Login User/Pass Scrub
    - Module: 1 - TweetExtractor.py
    - Description: Not scrubbing username and password from the Login page once entered.
    - Category: Security
2. Chrome Data Breach Popup
    - Module: Unknown, possibly 7- Login Page
    - Description: On entering a username/password combination and clicking submit, Chrome displays a popup that states there is a data breach.
    - Category: Security
3. Backend User Credentials
    - Modules: 4 through 21
    - Description: We need user credentials to be passed from the front-end process to the backend so that multiple users can access our application at the same time.
    - Category: Logic Error
4. Emoji Scrape Errors
    - Module: 1 - TweetExtractor.py
    - Description: If a tweet being scraped contained an emoji, its content was not able to be written to the CVS file. This could potentially be caused by not having an encoding type specified.
    - Category: Correctness
5. CSV Archive Data Loss
    - Module 1 - TweetExtractor.py
    - Description: The last few rows of data in the CSV file are cut off when extracted. This defect only takes place after the extraction of the CSV file.
    - Category: Logic Flaw
6. Category 2 Redundancy
    - Module 2 - InstagramURLKeywordExtractor.py
    - Description: Extra functionality was implemented allowing for scraping parameters that were not supposed to be implemented, specifically the scraping of user profiles. This was not supposed to be implemented and extra complexity that is not needed. This should be removed.
    - Category: Logic Flaw
7. Browser Waiting
    - Module 2 - InstagramURLKeywordExtractor.py

- Description: The InstagramURLKeywordExtractor uses the setting browser.implicitly_wait(5) when using selenium to navigate the headless browser. This setting makes the browser wait 5 seconds for a response when fetching something from a page. This should be changed to allow for slow internet so errors do not get thrown when they shouldn't be.
- Category: Convention

8. Sleeps in Code
   - Module 3 - PostExtractor.py and Module 2 - InstagramURLKeywordExtractor.py
   - Description: Sleep function calls were used in both of these files in spots where the code needed to do something, such as try catch blocks. These should be replaced with pass function calls.
   - Category: Logic Flaw
9. Comment Conventions
   - All Modules
   - Description: Comment conventions had not been created until the beginning of code inspection. This meant all code written was not following a standardized commenting convention and needs to be refactored.
   - Category: Comments
10. Code Conventions
    - All Modules
    - Description: Coding conventions had not been created until the beginning of code inspection. This meant all code written was not following a standardized coding convention and needs to be refactored.
    - Category: Convention
11. Fetch Requests deformed
    - Modules: 7 through 21
    - Description: Our fetch requests currently don't produce enough information for our restAPI to consume, limiting us from passing vital information such as user credentials. We need to modify the fetch requests to support all incoming data and information that the backend processes require.
    - Category: Logic Flaw
12. Login Auth Fails
    - Module 7 - LoginPage
    - Description: Our login page currently bypasses user authentication and gives access to the site without credentials. We need to render new pages conditionally to bypass this issue.
    - Category: Logic Flaw, Security
13. Database Displays
    - Module 5 - Endpoints
    - Description: Our database is currently displayed to everyone if a certain endpoint for the restAPI is hit regardless of credentials. This functionality needs to be masked or removed prior to release.

- Category: Security

14. Scrape Twitter Not Correctly Parsing
    - Module 6 - Endpoints
    - Description: When the front end passes information to the scrape twitter endpoint, it doesn't correctly parse it into the format specified when passing it into the Twitter query builder. This needs to be fixed to correctly parse the information.
    - Category: Logic Flaw
15. Scraping More Tweets Than Requested
    - Module: 1 - TweetExtractor.py
    - Description: Call to Twitter's API not returning the requested amount of tweets. In the case of this defect that was observed double the number of tweets were returned than requested.
    - Category: Logic Flaw
16. Checking for User Already in Database
    - Module 5 - Database Schema
    - Description: Our check to see if a user is already in the user database prior to account creation doesn't happen. This needs to happen or we will have duplicate user information within the database which leads to faulty coding and ambiguity.
    - Category: Logic Flaw
17. Deployment Pipeline Breakage
    - Module 23 - Pipeline
    - Description: Our deployment pipeline needs to be updated with new dependencies that the team is using so that we can deploy our software to an AWS instance and release it to our client.
    - Category: Logic Flaw

Appendix A - Coding and Commenting Conventions

The following information defines the coding conventions Black Bear Analytics has taken to improve its product code by increasing readability and coding efficiency. This is a comprehensive list of all standards the team has set and are agreed upon by the whole team. During code review, these standards will be used to define convention errors for both code and comments. Note that the symbol '<-' and anything after it are there for the reader's understanding of the convention and are note part of the convention itself.

Conventions for Variables

Variables in Python must start with the type of data that the variable will hold. This followed by an underscore and a meaningful name representing what is stored in it. Note that the name is using snake conventions, where _ separates words. Example:

```
l_links <- representing a list that contains links
re_email_contact <- representing a regex to match emails
```

Variables in React must start with the typeof data that the variable will hold. This is followed by an underscore and a meaningful name representing what is stored in it. Note that the name is using camel-case conventions, where each new word is uppercased. Example:

```
i_thisVariable <- representing an integer variable
```

Specifically for React tag attributes, we define their names following the React variable naming convention. Example:

```
className = "s_loginContainer" <- where the keyword is at the end
```

Commenting Conventions

When commenting Python functions, triple quotes are used to wrap the comment and will be the first lines within the function. Included in this comment must be: what the function accomplishes (Description), explained parameters (Arguments), and the output of the function (Output). Example:

```
def InstagramURLExtractor():

        """
        Description:

        Arguments:

        Outputs:
        """
```

When commenting React functions, a multiline comment is used and will be placed above the function. Included in this comment must be: what the function accomplishes (Description), explained parameters (Arguments), and the output of the function (Output). Example:

```
/*
* What it accomplishes:
*
* Arguments:
*
* Output:
*
*/
handleHello() {
        console.log(“Hello World!”);
}
```

Comments for Classes

In Python, comments at the top of classes and at the top of files must start and end with triple quotes and also who created it initially, the date of creation, the version, and detail what the class does. Example:

```
"""
Name: Abdul Karim
Date Created: 01/01/01
Version: 1.0
Description: foobar
"""
```

In React, comments above classes and the top of files must be wrapped in a multi-line comment and detail who created it initially, the date of creation, the version, and what the class does. Example:

```
/*
* Name: Abdul Karim
* Date Created: 01/01/01
* Version: 1.0
* Description: foobar
*/
```

In CSS, comments at the top of files must be wrapped in a multi-line comment and detail what page the CSS is for, who created it initially, the date of creation, and the version. Example:

```
/*
* Name: Abdul Karim
* Date Created: 01/01/01
* Version: 1.0
* Description: foobar
*/
```

Inline Comments

Python inline comments must be placed above the line of code it is documenting. Comment must start with a “#” followed by a single space and then a meaningful description of what the code accomplishes. Example:

```
# Object foo calls bar
o_foo.bar()
```

When first initializing a variable in Python, it is appropriate to have the inline comment to the right of the assignment. Example:

```
i_bar = 0          # This is the bar, initialized to 0
```

React inline comments must be placed above the line of code it is documenting. Comment must start with “//” followed by a single space and then a meaningful description of what the code accomplishes. Example:

```
// Object foo calls bar
o_foo.bar()
```

When first initializing a variable in React, it is appropriate to have the inline comment to the right of the assignment. Example:

```
i_bar = 0          // This is the bar, initialized to 0
```

When commenting in React’s render function any comment must start with “{/*” and end with “*/}”. Example:

```
{/* Object foo calls bar */}
o_foo.bar()
```

Function and Class Naming

When naming functions within Python, the return type of the function must be specified with a character followed by an underscore and then the function name. Example:

```
def i_function_name(): <- This function returns an integer
        return 0
```

In the case where the function will be for internal use only, the first character must be a single underscore followed by the described convention above. Example:

```
def _s_function_name(): <- This function is internal and returns a string.
```

```
        return “foobar”
```

When naming functions within React, the return type of the function must be specified with a character followed by an underscore and then the function name. After the functions parameters, there must be a single space before the open curly bracket. Example:

```
i_function_name() { <- This function returns an integer
        return 0
}
```

In the case where the function will be for internal use only, the first character must be a single underscore followed by the described convention above. Example:

```
_s_function_name() { <- This function is internal and returns a string.
        return “foobar”
}
```

Appendix B – Team Review Sign-off

All members of Black Bear Analytics have gone through the code review process and agree on both the defined coding conventions in this document and the code review defect results. We have also received a sign-off from the client, approving this document which means they have reviewed and have accepted what is listed above until fixed.

Name	Signature	Date
Customer:		
____________________	____________________	____________________

Comments:

Team:

Ryan M. Handlon	[illegible]	March 15th, 2021
Comments:		
Abdullah I. Karim	[illegible]	March 15th, 2021
Comments:		
Griffin L. Fluet	Griffin L. Fluet	March 15th, 2021
Comments:		
Colleen DeMaris	Colleen DeMaris	March 15th, 2021
Comments:		
James West	James West	March 15th, 2021

Appendix C - Document Contributions

Abdullah, Colleen, Griffin, James, and Ryan all contributed equally (20%) to this document. Each member showcased their code for review while others provided egoless comments on how to improve the code. In drafting this document each member listed defects they had, defects that were found during review, and commenting conventions that improve the readability and efficiency of our code. Each member also contributed to the refactoring of product code in light of defects and underutilized coding conventions.

Appendix E - AM

Product: S.S. Media

Client: Doctor Judith E. Rosenbaum

Administrator Manual

BLACK BEAR ANALYTICS

Black Bear Analytics

Abdullah Karim | Colleen DeMaris | Griffin Fluet | James West | Ryan Handlon

April 7th, 2020

Revision History

Version Number	Release Date	Description
Version 1.0	04/07/2020	Original Release

S. S. Media

Code Inspection Report

Table of Contents

S. S. Media Abstract

Social media has grown to be an extremely large part of society in recent years. Because of this, a need to gather data from it has also arisen. Black Bear Analytics is creating a tool that allows researchers to scrape public data from posts on Twitter and Instagram. This tool was requested by Doctor Judith Rosenbaum of UMaine's Department of Communication and Journalism. The S.S. Media will have an easy-to-navigate interface that allows users to switch between scraping public posts on Instagram and Twitter. The *New Search* page gives the user an option of either an advanced search or a basic search. The basic option searches by specified hashtags, locations, or phrases, and an acceptable start and end date to check with each post scraped, while an advanced search has the same general functions as the basic search, but adds the ability to search for more than one topic (hashtag, location, phrase), as well as run the search once every specified amount of time. The tool will store a number of scrape requests, so that users can either inspect a previous scrape request or request it again. Once the scraping is completed, the data will be downloaded to the user's computer in the form of a .csv file, and images will be stored in a corresponding folder. The outcome of this project will hopefully be that researchers like Doctor Rosenbaum can have a tool that will enable them to easily search for data and trends on social media sites.

1. **Introduction**

The purpose of this section is to introduce the reader to both this document and the Administrator Manual, detailing introductory information that will be useful for the reader to know.

1.1 Purpose of This Document

The purpose of this document is to inform the administrator of the S.S. Media Social Scraper about the system. This document will provide basic information on the system as well as any relevant information that should be known to maintain the system. This includes things like a system overview, hardware and software requirements, administrative procedures such as routine maintenance tasks, periodic maintenance tasks, and user support, and troubleshooting procedures such as known issues and how to handle them.

1.2 References

Black Bear Analytics SRS Document

Black Bear Analytics SDD Document

Black Bear Analytics CIR Document

Sofija SimicSofija Simic is an aspiring Technical Writer at phoenixNAP. Alongside her educational background in teaching and writing, & Sofija Simic is an aspiring Technical Writer at phoenixNAP. Alongside her educational background in teaching and writing. (2021, March 3). *How to Install Docker on Raspberry Pi (Step-by-Step Guide)*. Knowledge Base by phoenixNAP. https://phoenixnap.com/kb/docker-on-raspberry-pi.

Thiel, R.-M. (2021, January 24). *Setup your Raspberry Pi for Docker and docker-compose*. https://pumpingco.de/blog/setup-your-raspberry-pi-for-docker-and-docker-compose/.

2. **System Overview**

This section details background information on how the system works and what the system administrator would need to get started with our application. It also details hardware and software requirements that an administrator would need to set the system up.

2.1 Background

The S.S Media system being maintained will be a web application hosted on a Raspberry Pi. The front end, back end, and database will all be hosted on that pi and will be mostly self-contained. The administrator won't have to deal with them much after the initial setup. The administrator will mostly need to manage the administrator account for the application and various accounts. This includes things like approving user accounts, answering user queries, maintaining the Twitter developer account, and making sure the Twitter developer account and application's domain name are paid for. There won't be much day-to-day maintenance for the administrator other than making sure the application is still running and keeping up with user approval.

For this system, the backups are limited to a few config files and the database itself. Flask SQLAlchemy, the python library we use for database access, creates a file that becomes our database. We would need to back this file up. To do so, we include a script called backup.py that can be run from a terminal with the python interpreter which overwrites the saved database file on the USB with a new one every time a new entry is made to the database. The file will also back up the current config files used to scrape Twitter and Instagram called TwitterConfig.py and InstagramConfig.py.

> **WARNING:** TwitterConfig.py and InstagramConfig.py SHOULD NEVER BE ACCESSIBLE BY THE PUBLIC. They contain sensitive information like the user database does. Exposing the config files would be a security threat.

In case of a full system crash, the code uploaded on GitHub can act as a recovery for the system as well as the database entry and the config files in the USB. Follow the steps under section 3.3 to get full recovery details. In case the physical hardware needs to be relocated, shut down the hardware, relocate it, and reboot. The program should start automatically in the background.

2.2 Hardware and Software Requirements

The only hardware requirement for this is a Raspberry Pi 4. The Administrator will need to set up this minicomputer in a cool, temperature-controlled environment to ensure that it does not overheat while running. A fan or another way to cool the Raspberry Pi is also necessary. A backup drive of about 2 GB or greater should be inserted into the Raspberry Pi to serve as a backup for the system.

There are many software dependencies that our system depends on to get the application going. To start, the Administrator will have to make sure the Raspberry Pi is equipped with the Raspian OS or Ubuntu 18.04. The Administrator will then have to download Python, Docker, and install the USB backup drive. A stable internet connection will also be required for spinning up the program.

3. **Administrative Procedures**

This section details the day-to-day and periodic operations an administrator should perform if any. It also details the installation process and what to do in case of an emergency.

3.1 Installation

The system will be preinstalled on to a Raspberry Pi 4. The first time the application is run the database will need to be created via terminal and populated with an administrator account. Three files will also need to be obtained, these are TwitterConfig.py, InstagramConfig.py, and Chomedriver. The two configuration files are needed for secure credentials and tokens. These files should not be shared over the internet in any way. Chromedriver is needed for Instagram scraping to emulate a web browser. These files will be stored on the backup flash drive and be periodically updated with the existing files in our application. For whatever reason a fresh install of the application should need to be done the following steps should be followed:

To access the host machine via an external source, the terminal (for Mac) or command prompt (for Windows). The command ssh {username}@{host ip} -p 532 should be specified to gain access to the pi. From there, all files need to be transferred to the pi ugithubsing the getResources.py script within the software's directory. This will move all required resources to the pi from the installed external storage unit.

We utilize software called Docker to create instances of the front and back end of our application that the user can then see. Rather than specifying instructions that can become stale in the future, we provide a link to the resources required to install Docker, which is maintained by external sources and should not be stale information. To install Docker on the host machine (i.e. the raspberry pi) follow this link:

https://phoenixnap.com/kb/docker-on-raspberry-pi.

This page will allow you to set up Docker on the hosting machine to run our application. Docker-compose will also need to be installed on the system to generate each container. The following link is a walkthrough of how to install docker-compose:

https://pumpingco.de/blog/setup-your-raspberry-pi-for-docker-and-docker-compose/

Once Docker and docker-compose are installed and all files are installed on the machine, navigate to the installation directory's root folder and perform a "docker-compose up". This will start both the back-end and front-end services for the user and will launch our application.

3.2 Routine Tasks

The administrator will be responsible for approving new accounts when a new account request is made. This is done by going to the settings page, finding the account, and clicking the 'Approve' button.

3.3 Periodic Administration

General Periodic Administration:
The administrator will have an email account that will be used for approving accounts when needed. Users are also able to send questions and concerns through the web application to the administrator via this email. The administrator can ban any account that is on the application if the account is no longer active.

Twitter Periodic Administration:
The administrator will also have access to the Twitter Developer Portal. Since you have to pay monthly for the requests when using the Twitter API, the developer portal allows the administrator to monitor the request use. The developer portal also gives the ability to check when the subscription will run out and where you can change subscription tiers. From the portal, the administrator can also generate new tokens that are used to access the API. If the tokens were to ever be exposed, regeneration of the API tokens is required. Part of the paid tier API also gives access to an exhaustive search of every tweet posted since Twitter's launch. To gain access to this search feature a dev environment must be created. Both the tokens and the name of the dev environment need to be added to the TwitterConfig.py file. This file will live on the Raspberry Pi within the twitter directory of the application. Changes to this file will have to be made through ssh and using a command-line text editor such as nano. There are six variables within this file that correspond to what they hold, s_consumer_key, s_consumer_key_secret, s_access_token, s_access_token_secret, s_bearer_token, and s_dev_environment. Copy the new tokens to the existing variables and save the file.

Instagram Periodic Administration:
InstagramConfig.py is a configuration file used to provide necessary information to the Instagram scraping process. A few of these pieces of information will be needed to be changed from time to time.

The first is s_path_to_driver. During setup, this value needs to be set to the location where the chromedriver is stored on the Raspberry Pi. If the chromedriver is moved, this value will also need to be changed to accommodate that. The value should be surrounded by quotes to represent a string and look like: s_path_to_driver = “~/path/to/driver”.

The next thing that likely needs changing is the cookie value in d_headers. This value is used to help configure the browser used to scrape posts. If for any reason Instagram scraping breaks, this is the first thing you should look into. You can update this value very easily. To do so, login to instagram.com in any account on google chrome. Next, click the f12 button on your keyboard. At the top, there will be a bar with options such as elements, console, sources… Select Application. There will be a bunch of options on the left side of the window. Select cookies, then the https://www.instagram.com option. This will bring a bunch of values such as ig_did, mid, and ig_nrcb. In InstagramConfig.py, the id_headers value will be set equal to a value such as: “ig_did=RadomInformation; mid=RandomInformation;...”. For each random information value, replace it with the new one that can be found on the Instagram cookies-page you opened.

InstagramConfig.py file also holds a dictionary of Instagram usernames and associated passwords. Although this hopefully shouldn't happen, there is a chance that these accounts get timed out or banned. If for any reason Instagram scraping breaks, this is the second thing you should look into. Should this happen the administrator will need to create new accounts by going to Instagram.com and adding those account credentials to the InstagramConfig.py file. The dictionary should look like this:

```
d_accounts = {“username”: “password”, “username”: “password, “username”: “password”}
```

Backup Periodic Administration:
Backups will be done through cron scheduling and will happen automatically. The backup process is initiated every 6 hours, or whenever an entry to the database is created. Backups of the user database, TwitterConfig.py, and InstagramConfig.py will be made and stored on the backup flash drive attached to Raspberry Pi.

3.4 User Support

The system allows existing users and new users to contact the administrator by email from the Contact Us page. On the Contact Us page, the user will need to enter their email address along with the message they want to send to the administrator, these can be up to 140 characters long. Existing users can use this contact method to alert the administrator of any potential bugs or errors. The support account credentials are added in a text file on the external storage unit.

4. **Troubleshooting**

4.1 Dealing with Error Messages and Failures

If an error has occurred, it is best to ensure that the information you are attempting to put in matches the exact format that the input is asking for. If the error or failure results in an error screen, first attempt to go to the Home Page link. If the Home Page also results in an error screen, then the Raspberry Pi will need to be rebooted.

If the error or failure results in a screen that has bizarre formatting, stretched components, or out of alignment components, try resizing your window or changing your screen resolution settings. The system was tested most regularly on a 2880 x 1800 resolution using Google Chrome as the web browser.

If the error or failure results in a Twitter scrape ending without any data having been collected, ensure that the scrape criteria are correctly formatted. The next step is to check if the Twitter API subscription is current and still has not hit its data limits.

If the error or failure results in an Instagram scrape ending without any data having been collected, ensure that the scrape criteria are correctly formatted.

Dealing with Serious Errors

NOTE: It's best to ensure that all dependencies (i.e. backup files on the external storage unit) are installed onto the machine before debugging any errors.

For back-end development work, open the project in a code editor, and open a terminal (We recommend VSCode as a code editor). Use the 'cd' command to get into the **API** directory. All dependencies for both the frontend and backend processes are specified in the Dockerfile associated with that directory. The Dockerfile for the frontend is located in the my-app directory while the Dockerfile for the backend is located in the API directory. Should any more dependencies for the backend need to be installed, simply add a command to the Dockerfile under API in the following form:

RUN [Insert command here] (ex. RUN pip install selenium)

This will install a new dependency into the container, which will be installed and run when "docker-compose up" is called. To see exactly what is going on behind the scenes, the comments within each Dockerfile go in-depth on what each part of the process is doing, so following those detailed comments is the best way to understand the container structure of our application. This will help if there is an error on the initial startup.

4.2 Known Bugs and Limitations

This section is adapted and updated from the Code Inspection Report (CIR) referenced above. Below is a list of known bugs and limitations that we have found in our code:

1. **Input Bug**
    a. **Impacts:** SearchCriteriaPage.js

   b. **Description:** When inputting into the HashTags, Locations, and Phrases input boxes, the last character is not stored in the input variable. This causes an input of '#minecraft' to be '#minecraf' instead.
   c. **Constraints on Fixing:** Time limitations. This will be fixed by changing how the state is being saved.
   d. **If encountered:** This will not be encountered by a user. If encountered, simply add a space as the last character for inputs.
2. **Styling Issues**
   a. **Impacts:** All Pages
   b. **Description:** There is very little CSS written to help ensure pages format correctly to other screen sizes. This results in buttons and other components being out of alignment in comparison to the mock-up's placement for untested screen resolutions.
   c. **Constraints on Fixing:** Time limitations. CSS takes a while to play with and get styling just right for a user. Try editing the CSS of the project to fix the framing issues.
   d. **If encountered:** Resize the browser window used to access the website until it looks good.
3. **Chrome Data Breach Popup**
   a. **Impacts:** Login Page
   b. **Description:** On entering a username/password combination and clicking submit, Chrome displays a popup that states there is a data breach.
   c. **Constraints on Fixing:** Time constraints. More research needs to be done on this to understand the cause of google chrome flagging our backend as a data breacher.
   d. **If Encountered:** The username and password are not breached. Ignore the message if it arises, our backend system is just capturing the input information from the frontend.
4. **CSV Archive Data Loss**
   a. **Impacts:** TweetExtractor.py
   b. **Description:** The last few rows of data in the CSV file are cut off when extracted. This defect only takes place after the extraction of the CSV file.
   c. **Constraints on Fixing:** Time constraints. More research needs to be done here. A potential fix is after opening a CSV file, save it as an XLSX file.
   d. **If encountered:** Understand that the information provided is missing a few data points and potentially input a scrape that would grab those.
5. **Instagram Headers becoming outdated**
   a. **Impacts:** PostExtractor.py
   b. **Description:** The headers that provide information about the browser can become outdated. When this happens the headers will no longer work, making

the PostExtractor malfunction. The PostExtractor is an integral part of the Instagram scraping process, and with that not working, the Instagram scraping process won’t work.

   c. **Constraints on Fixing:** This issue wasn’t seen far enough in advance and would require man-hours when we were working on time constraints.
   d. **If Encountered:** If this bug is encountered, follow the associated procedure shown in section 3.3 Periodic administration under the Instagram Periodic Administration subsection

6. **Instagram Accounts getting Banned**
   a. **Impacts:** InstagramKeywordURLExtractor.py
   b. **Description:** There are cases where an Instagram account will get banned. If enough accounts get banned, the InstagramKeywordURLExtractor will not be able to log into accounts to scrape posts. This will make it so Instagram posts can no longer be scraped.
   c. **Constraints on Fixing:** In our current implementations, due to restrictions by Instagram.com, the bug isn’t fixable.
   d. **If Encountered**: Should enough Instagram accounts get banned to the point where scraping is no longer possible, follow the associated procedure in section 3.3 Periodic administration under the Instagram Periodic Administration subsection

Appendix A - Team Review Sign-off

All members of Black Bear Analytics have gone through the review process for this document and agree on both the manual entries in this document and are certain there is enough coverage to be effective. We have also received a sign-off from the client, approving this document which means they have reviewed and have accepted what is listed above until fixed.

Name	**Signature**	**Date**
Customer:		

Comments:

Team:

Name	Signature	Date
Ryan M. Handlon	[illegible]	April 7th, 2021

Comments:

Name	Signature	Date
Abdullah I. Karim	[illegible]	April 7th, 2021

Comments:

Name	Signature	Date
Griffin L. Fluet	Griffin L. Fluet	April 7th, 2021

Comments:

Name	Signature	Date
Colleen DeMaris	Colleen DeMaris	April 7th, 2021

Comments:

Name	Signature	Date
James West	[illegible]	April 7th, 2021

Appendix B - Document Contributions

Abdullah, Colleen, Griffin, James, and Ryan all contributed equally (20%) to this document. Each member added to a section(s) of the document with valuable content. In drafting this document each member listed defects they had, solutions for problems, and general information to help an administrator run S.S. Media.

Appendix F - UM

Product: S.S. Media

Client: Doctor Judith E. Rosenbaum

User Manual

Black Bear Analytics

Abdullah Karim | Colleen DeMaris | Griffin Fluet | James West | Ryan Handlon

April 19th, 2020

Revision History

Version Number	Release Date	Description
Version 1.0	04/19/2020	Original Release

S. S. Media
User Manual

Table of Contents

S. S. Media Abstract

Social media has grown to be an extremely large part of society in recent years. Because of this, a need to gather data from it has also arisen. Black Bear Analytics is creating a tool that allows researchers to scrape public data from posts on Twitter and Instagram. This tool was requested by Doctor Judith Rosenbaum of UMaine's Department of Communication and Journalism. The S.S. Media will have an easy-to-navigate interface that allows users to switch between scraping public posts on Instagram and Twitter. The *New Search* page gives the user an option of either an advanced search or a basic search. The basic option searches by specified hashtags, locations, or phrases, and an acceptable start and end date to check with each post scraped, while an advanced search has the same general functions as the basic search, but adds the ability to search for more than one topic (hashtag, location, phrase), as well as run the search once every specified amount of time. The tool will store a number of scrape requests, so that users can either inspect a previous scrape request or request it again. Once the scraping is completed, the data will be downloaded to the user's computer in the form of a .csv file, and images will be stored in a corresponding folder. The outcome of this project will hopefully be that researchers like Doctor Rosenbaum can have a tool that will enable them to easily search for data and trends on social media sites.

1 Introduction

The S.S. Media is a social media data gathering platform created by the Black Bear Analytics (BBA) team from the capstone class (COS 497) at the University of Maine, from 2020-2021. Our customer is Doctor Judith E. Rosenbaum from the Communications and Journalism department at the University of Maine. BBA undertook this project to help researchers in the social sciences gather information from social media platforms such as Instagram and Twitter. To date, core functionality for these platforms has been created and this User Manual serves to help administrators, support staff, and users maintain the software and use it to access data.

1.1 Intended Readership

The personas that this User Guide and accompanying Administrator Manual recognize are covered in this section. BBA has compiled a list of users, and descriptions of their function in maintaining or using S.S. Media.

In the following section, we go over each persona, a description of that persona, the level of experience assumed, and sections of the manual that are most relevant to them:

- Persona 1: Administrator
    - Administrators are responsible for user management. They are given the power to approve new users, ban user accounts, and delete users.
    - Administrators are expected to understand minimal technology concepts and are capable of performing functions and procedures listed in the Administrator Manual document.
- Persona 2: End User
    - The end user will be the person using the product. They have full access to scraping information through the search page. They do so with keywords from websites or links to locations presented on the website.
    - End users are expected to have basic computer skills such as typing and operating a web browser like Google Chrome. No technical experience is expected.
- Persona 3: Developer
    - This persona is a helper to the product. They maintain and push updates alongside maintenance features to the product when requested or required.
    - Expected to have a full understanding of technological methodologies as well as programming languages and software development skills. Able to troubleshoot, research, and fix underlying issues within the code of the tool.
- Persona 4: Support Outreach
    - Support Outreach deals with end user contact from the contact page. They are also responsible for keeping up a subscription with the Twitter API and any other budgetary need pertaining to the product.
    - Basic computer skills and knowledge of performing online purchases, such as

signing up and cancelling subscriptions is expected.

This User Guide applies to S.S. Media Version 1.0.

1.2 Purpose

The purpose of the system we are designing is to take key search terms and understandings from the user and process it into meaningful social media posts related to the search terms that help the user conduct research on social media topics.

This process is the research process that many researchers at universities and businesses undergo to get data from social media sites. The data is collected manually or by miniature scripts in today's world, but BBA is seeking to streamline the data collection process with an application that molds itself into a useful tool for data gathering.

1.3 How to Use this Document

This User Guide has three main sections along with subsections that flesh out its details. The main sections are: the Introduction, the Overview, and Instructions.

Section 1, the Introduction, introduces the reader to this document and developed application. This includes things such as intended readership, the purpose of the system, how to use the document, and related documents.

Section 2, the Overview, provides a brief overview of the application, providing the reader with a general idea of how the application works.

Section 3, the Instructions, provides detailed instructions on how to use the application. This includes a subsection dedicated to each page of the application. Each subsection provides a picture of what the page looks like, details the purpose of the page, and describes all parameters and functionality of that page.

1.4 Related Documents

Below is a list of all documents that are related or referenced by this document:

Table 1.1: Related and Referenced Documents

Num	Title	Author	Date	Issue
1	S.S. Media System Requirements Specification Document	Black Bear Analytics Team	10/24/2020	1.0
2	S.S. Media System Design Document	Black Bear Analytics Team	11/102020	1.0

3	S.S. Media User Interface Design Document	Black Bear Analytics Team	11/24/2020	1.0
4	S.S. Media Critical Design Review Document	Black Bear Analytics Team	12/17/2020	1.0
5	S.S Media Code Inspection Report	Black Bear Analytics Team	3/17/2020	1.0
6	S.S. Media Administrator Manual	Black Bear Analytics Team	4/7/2020	1.0

1.5 Conventions

This document has no important stylistic conventions.

1.6 Problem Reporting Instructions

If a user encounters any software issues, the best course of action is to contact the system administrator. This can be done through the Contact Us Page on the application. This will notify the system administrator via email so they can get in contact with the user to provide help. Expect a response email to the one provided on the Contact Us Page.

2 Overview

S.S. Media is a web application developed to scrape Instagram and Twitter for relevant posts, based on hashtags, locations, or phrases. It is accessed through a web browser, such as Chrome, Safari, or Internet Explorer. Upon visiting the site, users will be brought to the login page. Once logged in, they can navigate through the rest of the site. There are 10 total pages. They are the Login User Page, Register Account Page, Account Processing Page, Contact Us Page, Contact Us Success Page, Home Page, User Settings Page, Administrator Settings Page, New Search Page, and the Searching Page. With the web application, users will be able to create an account, login into their accounts, manage their settings, and scrape information off of the social media sites Twitter and Instagram. The administrator has additional powers, allowing them to approve, delete, and ban other user accounts. All users can scrape information based on different search parameters such as hashtag, location, and phrase. Once a scrape has finished, the information will be returned to them as a zipped folder containing a .csv file full of all scraped text information and a media folder containing scraped pictures and videos associated with scraped posts.

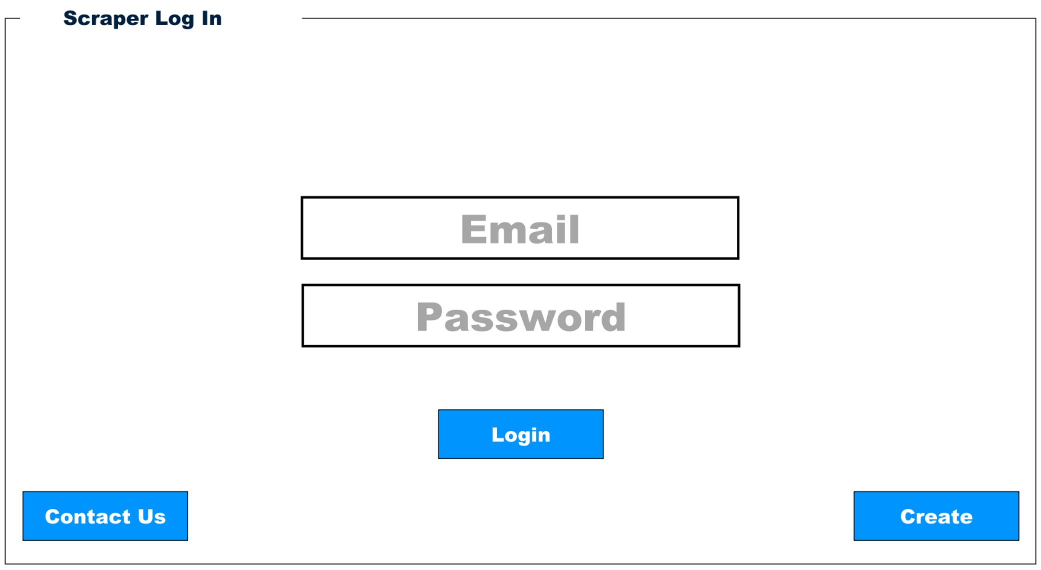
3. Instructions

The S.S. Media application has a user system that allows new visitors of the site to create an account, have the account be approved by an administrator, and login. Upon a successful login, the user will have access to the data scraping page that allows the user to enter criteria that the data being scraped must have. The criteria that can be validated against are dependent on the website being scraped. For Twitter, the user may enter locations or hashtags as criteria for what the data, or tweets, must contain. Instagram will only accept locations or hashtags as criteria. Once a search has been submitted, the data will be searched for and collected into a zip file or folder, depending on the website being scraped.

For any questions or concerns about administrator functions as responsibilities, please refer to the Black Bear Analytics Administrator Manual. This includes the important administrator topics as follows: Hardware and Software Requirements, Administrator Procedures, System Installation, Routine Tasks, Backup Procedures, User Support, Troubleshooting, Dealing with Error Messages and Failures, and Known Bugs and Limitations.

Now we begin by taking a deep dive into each page by analyzing the page, any warnings associated with it, instructions on how to use the page, and any errors that could be encountered with possible remedies.

Login User Page



Functional Description of Page

The Login User Page is the landing page when a user first visits the application. With this page, a user can message the administrator, create an account, and login to their account.

Cautions and Warnings

N/A

Procedures

This page includes 2 text boxes and 3 buttons. They function as follows:

- Email Textbox:

  The email textbox takes in email input for a user's account.

- Password Textbox:

  The password textbox takes in password input for a user's account.

- Login Button:

  Upon being clicked, the login button authenticates the user. If the submitted email and password submitted in their associated boxes match a user's account, the user will be logged in and navigated to the home page,

- Contact Us Button:

  The Contact Us Button navigates the user to the Contact Us Page

- Create Button:

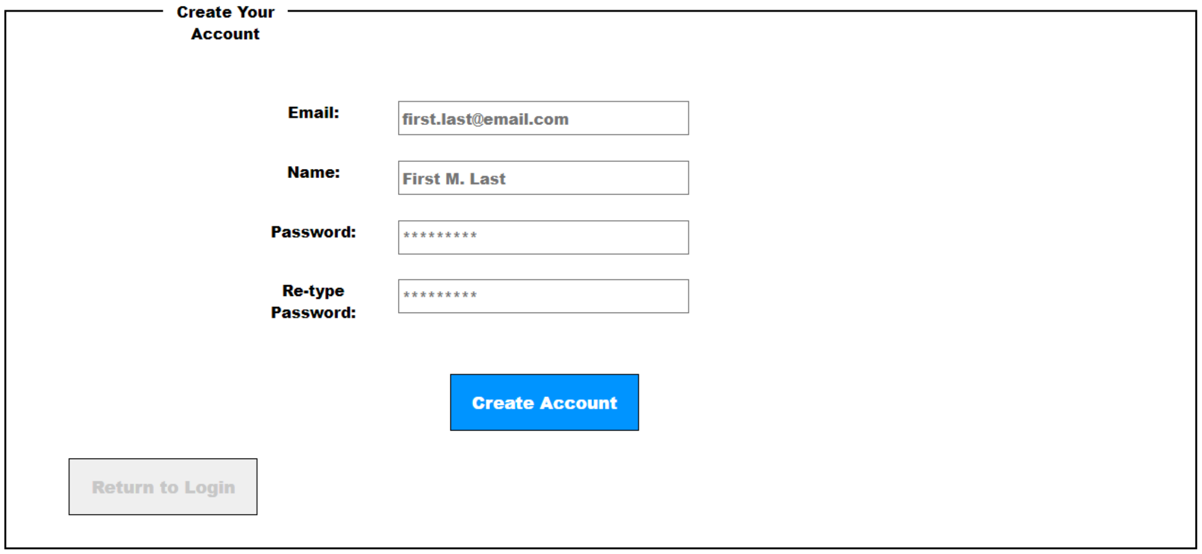
  The Create Button navigates the user to the Register Account page

Probable Errors and Possible Causes

- Not able to login:

  This may happen when a user enters their credentials wrong, when a user has not created an account, or if the account in question has been deleted or banned by an administrator.

Register Account Page



Functional Description of Page

This page allows a user to create a new account request. That request will be sent for administrators to view. Upon an administrator's approval the user's account will be created.

<u>Cautions and Warnings</u>

- There is no confirmation on the information being provided.
- It is not possible to reset a password for an account. Instead, in order to reset a forgotten password, the user must contact an administrator and give them their account's email address and inform them that they cannot access their account anymore. The account with the forgotten password will be deleted, and a new account must be created by the user.
- Leaving this page (such as refreshing or clicking "Return to Login") will remove any changes made to the text fields, and the user must start over.

<u>Procedures</u>

- From the LoginPage, click on the Create button found at the bottom right of the page.
- Fill out the form by clicking on each input box and typing the input as shown in the examples provided directly above each input box, respectively.
- Click on the Submit button at the bottom center of the page.

<u>Probable Errors and Possible Causes</u>

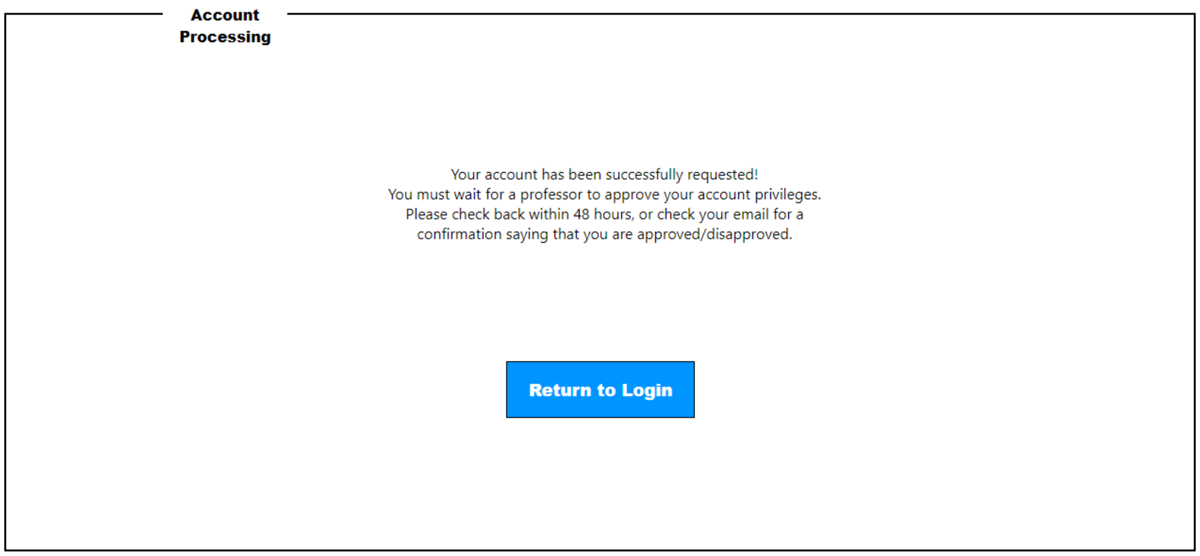
- Passwords do not match

  This error occurs when the "Password" and "Re-type Password" fields are filled out but not matching.

- Empty field

  This error occurs when any of the four given text boxes are empty when the "Create Account" button is pressed.

Account Processing Page



Your account has been successfully requested!
You must wait for a professor to approve your account privileges.
Please check back within 48 hours, or check your email for a confirmation saying that you are approved/disapproved.

Functional Description of Page

The Account Processing page is navigated to from the Register Account Page. This page's only purpose is to confirm to the user that their account was requested.

Cautions and Warnings

N/A

Procedures

This page has only one button, which upon clicking, will navigate the user back to the login page.

Probable Errors and Possible Causes

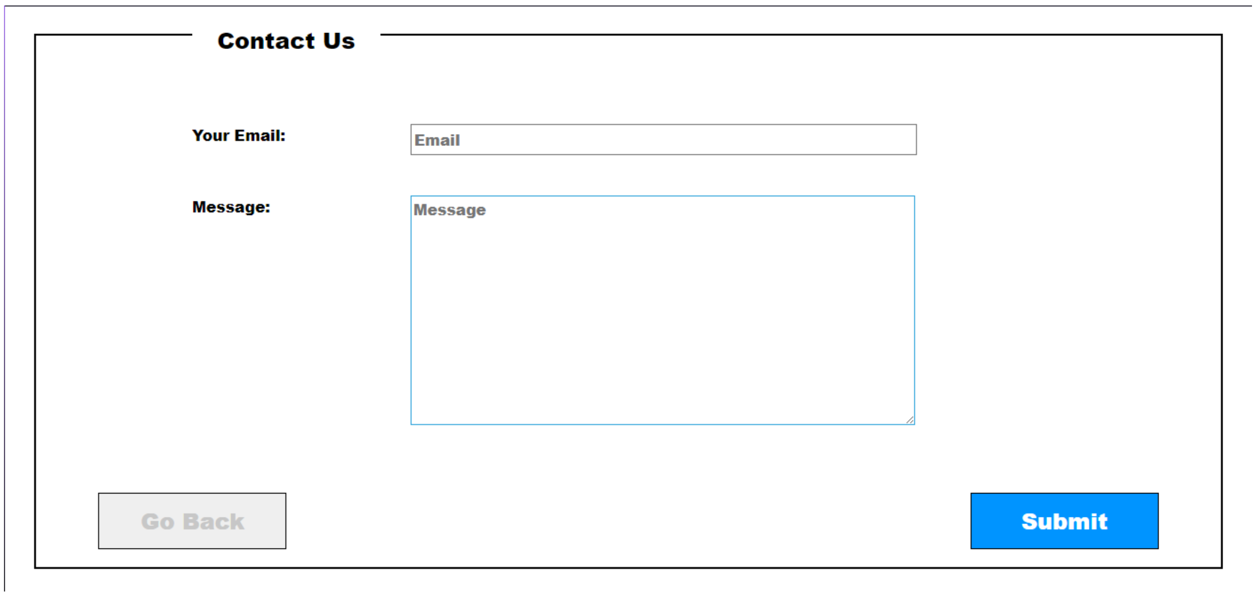
- The account was unsuccessfully created:

  This error might occur if there is a disconnect between the database and the front end, or if the database is down.

- An account with this email already exists:

  This error will occur if there is an account with the same email address already created.

Contact Us Page



Functional Description of Page

The Contact us page will allow a user to send any questions or concerns to the administrator by email. The user is asked to enter their email address and can send messages upto 500 characters long to the administrator. The administrator will respond to this message via the entered email address.

Cautions and Warnings

- No entered text will be saved or sent if the user leaves the Contact Us page before hitting the Submit Button.
- The message's information is not encrypted when sent, so do not send any extremely sensitive data over this contact portal.

Procedures

The Contact Us page includes 2 text fields and 2 buttons. They function as follows:

- Your Email Text Field:

  This text field allows for the user to enter their email address. An example of the entered text is supplied above the field. This is the email address that the administrator will send a reply to.

- Message Text Field:

  This text field allows for the user to enter their message to the administrator. Messages must be kept under 500 characters. Above the field is an indicator of how many characters left.

- Submit Button:

When the user clicks the Submit Button their message will be sent to the administrator. This will also bring the user to the Contact Us Confirmation Page to inform the user if the message has been successfully sent.

- Go Back Button:

  When the user clicks the Go Back Button they will be directed to the Login User Page.

<u>Probable Errors and Possible Causes</u>

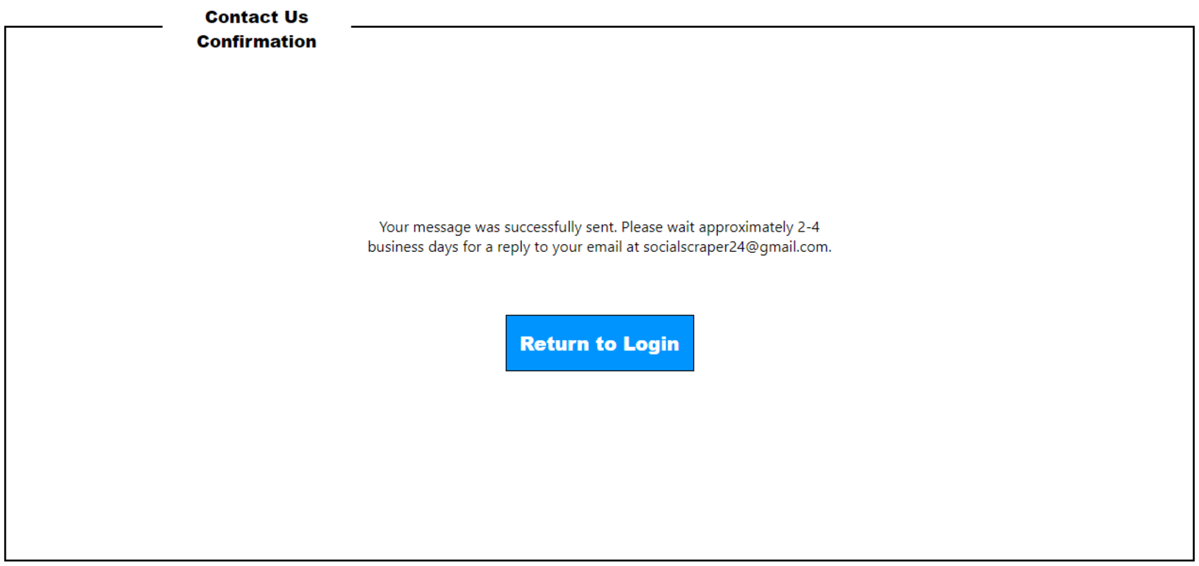
- Incorrect email format:

  This error is caused by a user not correctly writing their email account in the firstlast@email.com format.

- Too Many Characters:

  This error is caused by a user typing more characters than allowed 500 and attempting to send the message.

Contact Us Success Page



Your message was successfully sent. Please wait approximately 2-4 business days for a reply to your email at socialscraper24@gmail.com.

<u>Functional Description of Page</u>

This page will let the user know if their contact us message has been successfully sent. A message will be displayed depending on the outcome and a home button is shown to bring the user back to the Home Page.

<u>Cautions and Warnings</u>

N/A

Procedures

The Contact Us Success Page includes 2 potential messages and 1 Button. They function as follows:

- Sent Successfully Message:

  This message is displayed when the user's message was successfully sent to the administrator.
- Failed to Send Message:

  This message is displayed when the user's message fails to send.
- Home Button:

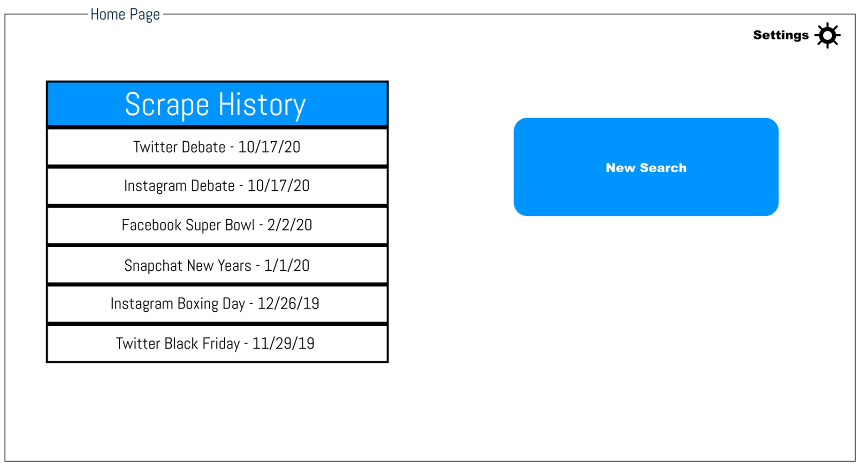
  When the user clicks the Home Button they will be directed to the Home Page.

Probable Errors and Possible Causes

- The message can be reported as unsuccessfully sent:

  The front end has lost connection to the script that sends the email, causing the message to not be able to be saved and sent to the administrator account.

Home Page


Home Page
Settings
Scrape History
Twitter Debate - 10/17/20
Instagram Debate - 10/17/20
Facebook Super Bowl - 2/2/20
Snapchat New Years - 1/1/20
Instagram Boxing Day - 12/26/19
Twitter Black Friday - 11/29/19
New Search

Functional Description of Page

The Home Page is used as a landing page for the user after logging in. With it the user can navigate to all features of the application. This includes viewing settings, creating a new search, and loading a previously saved scrape. The image above is the view of this page.

Cautions and Warnings

N/A

Procedures

The Home Page includes 3 buttons and one table. They function as follows:

- New Search Button:

  The New Search Button navigates the user to the New Search Page.

- Settings Button:

  The Settings Button navigates the user to the Settings Page

- Scrape History Table:

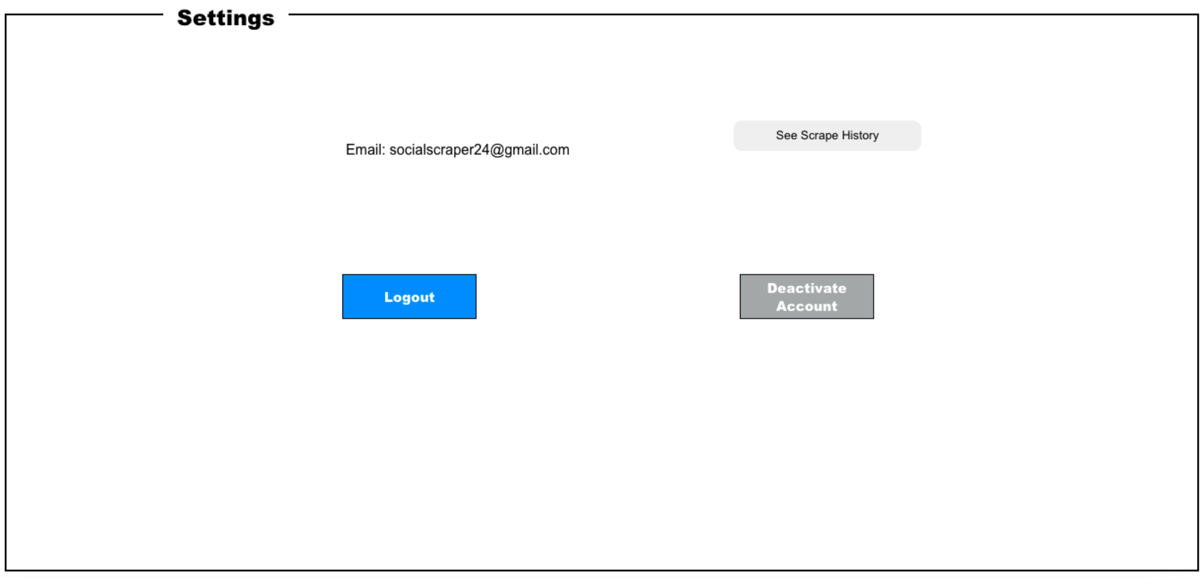
  The Scrape History Table holds references to the last 5 scrapes done by the current user. Clicking one of these references will navigate the user to the New Search Page with the parameters said search already filled in.

Probable Errors and Possible Causes

- Scrape history does not populate the New Search page correctly.

  This could be caused by the user running a scrape with an error in input and it getting saved to the user's scrape history.

User Settings Page



Functional Description of Page

The User Settings page is for non-administrative users to change their settings and save them to their account. The display includes the user's email address and the See Scrape History button that allows them to toggle whether they can see their last five scrapes on the home screen. The Logout button will log the user out of their account and redirect them to the home page. The Deactivate Account button will log the user out, remove the user's account from the system, and redirect them to the Login Page.

Cautions and Warning

- Deactivating the user's account will permanently and instantly deactivate the account. The account cannot be recovered afterwards, and a new account must be created by the user.

Procedures

There are 3 buttons and one text display field on this page. They function as follows:

- Email text display field:

  The user's email address they use to receive emails and log in is displayed here. They cannot change it, and instead must create a new account if they want a different email.

- See Scrape History:

  The user can toggle this button on and off, with the BLUE color on the button being ON and the WHITE color on the button being OFF.

When toggled to ON, the user will see the scrape history of the past five scrapes that they started, displayed on the home screen.

When toggled to OFF, the home page will not display any previous scrapes.

- Logout:

  Pressing this button will log the user out of their account and redirect them to the home page.

- Deactivate Account:

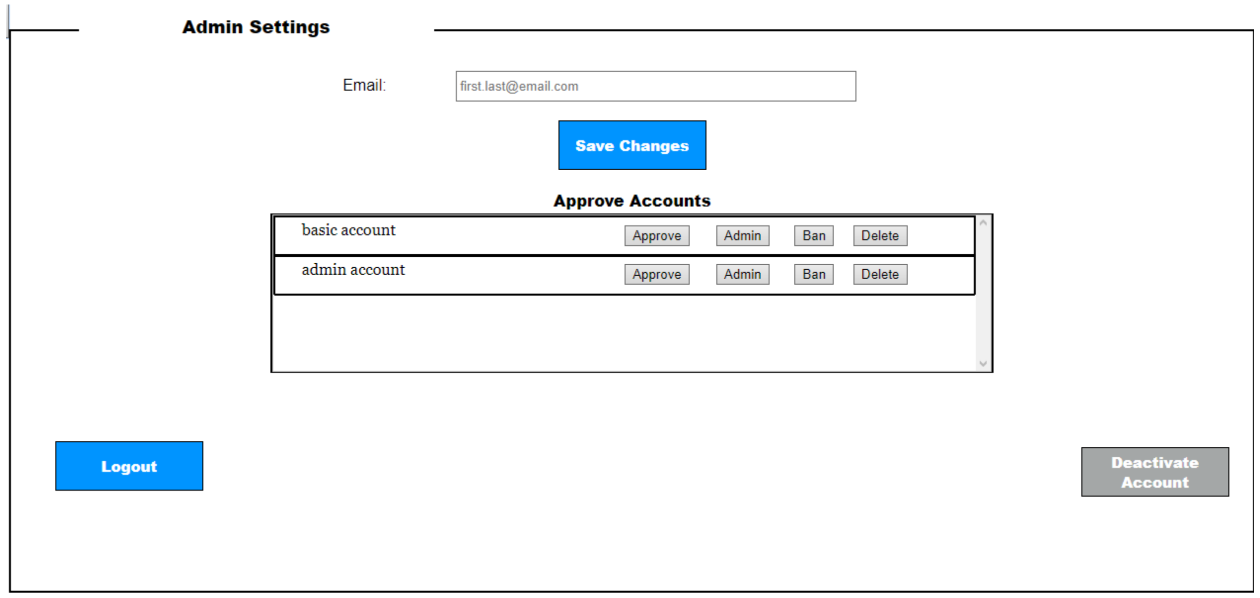
  Pressing this button will log the user out of their account, delete their account from the database, and redirect them to the Login page.

Probable Errors and Possible Causes

- Failure to Deactivate Account:

  This error might occur if there is a disconnect between the database and the front end, or if the database is down, or if the program returns that the email address that the user has is not in the database.

Administrator Settings Page



Functional Description of Page

The Administrator Settings Page allows the user to have access to the same functionality as the User Settings page, with the added functionality of approving, banning, deleting, and promoting accounts.

Cautions and Warnings

- There is no way to demote an administrator back to a user account, once the Admin button is clicked on an account, that account will always be an administrator.
- There is no way to unapprove an account, however you can still ban or delete any account.
- Do not ban or delete every administrator account or no admin accounts will exist without a way to add a new administrator or approve new accounts.
- There is nothing preventing a user from deleting their own account via the “Approve Accounts” section: it will have the same functionality as clicking the “Deactivate Account” button.

Procedures

The Admin Settings Page has 3 buttons and 1 text field. This page also has a list of all the active users, on the listed account there are 4 buttons. They function as follows:

- Logout Button:

  When clicked, the logout button will log the user out of the application and redirect them to the Login page.
- Save Changes:

  When clicked, this button will update the user's email if they are entered into the corresponding text fields
- Deactivate Account:

  When clicked, this button will deactivate the current user’s account.
- Email Text Field:

  This text field is for updating the email address associated with the user’s account.
- Active User List:

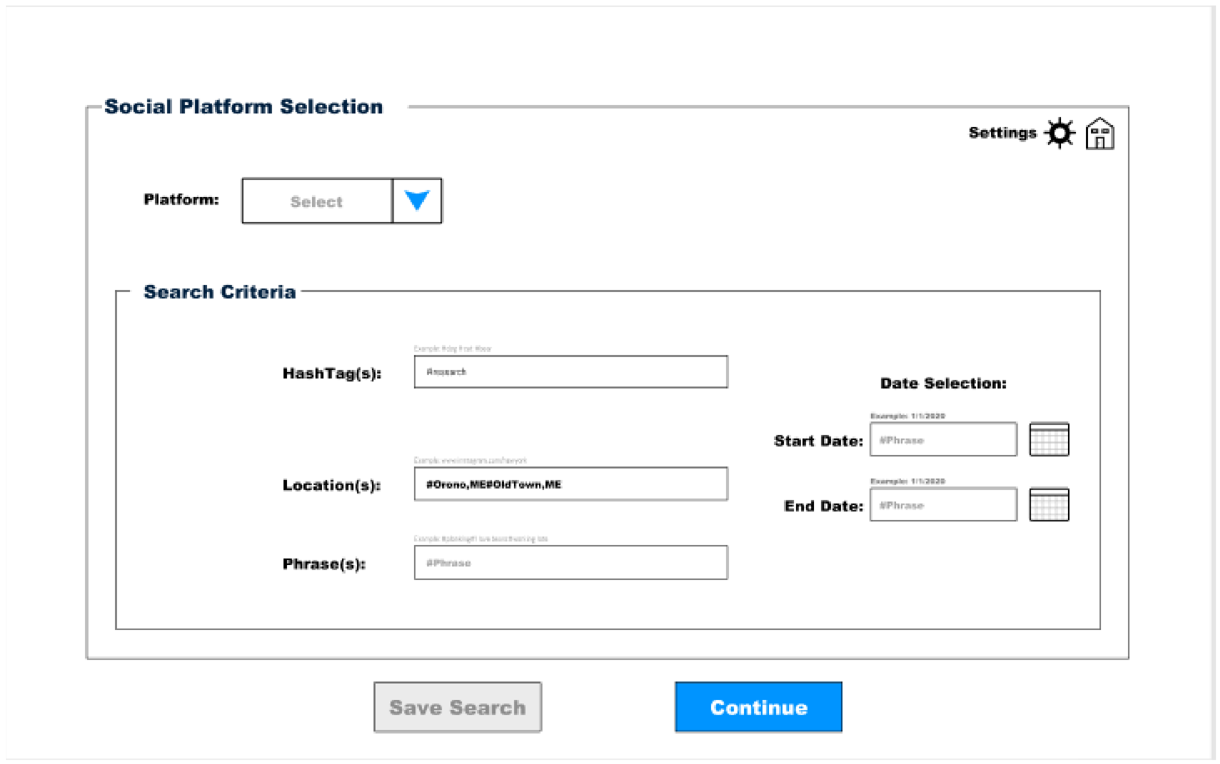
  This is a list of all the active users, each user gives the admin 4 options of buttons.
  - The approve button is for approving a pending user, granting them access to the application.
  - The admin button gives the user admin permissions, which gives them access to the admin settings.
  - The ban button is used for banning an account. This action keeps the user’s email in the database and will not allow a new account to be made with that email.
  - The delete button is used for deleting an account. Unlike the ban button this will also remove the email from the database. Allowing a new account to be made from that email.

Probable Errors and Possible Causes

- Failure to Ban Account, Failure to Deactivate Account, Failure to Approve Account, Failure to Delete Account

  This error might occur if there is a disconnect between the database and the front end, or if the database is down, or if the program returns that the email address that the user has is not in the database.

New Search Page



Functional Description of Page

The New Search page is used to help a user start a new social media scrape. It can scrape posts from both Twitter and Instagram and can also take search parameters such as hashtags, locations, and phrases. When a new scrape is started it will navigate the user to the Searching page.

Cautions and Warnings

Make sure your input is valid. There are many rules specified below on the valid inputs you can use. Using invalid input could break something. If unsure if your input is correct, consult the descriptions below.

Procedures

The Search Criteria Page contains 1 drop down menu, 5 text fields and 3 buttons. They function as follows:

- Platform Drop Down box:
  The Platform Drop Down Box had two options: Twitter and Instagram. Selecting on will show the available options to search by for that social media.
- Hashtags Textbox:
  <u>Instagram</u>
  If Instagram is selected, the textbox will take in a single hashtag. This hashtag will be the one you wish to search by. The hashtag inputted should be preceded by a hashtag. The input should look something like this: #tacos
  <u>Twitter</u>
  If Twitter is selected, the textbox will take in a list of hashtags. These hashtags will be added to the search parameters so that only tweets with that hashtag will be returned. Each hashtag inputted should be preceded by the hashtag symbol (#). The input should look something like this: #dog#cat#snake
- Locations Textbox:
  <u>Instagram</u>
  If Instagram is selected, the locations textbox will take in a link to an instagram location page. This will be the location you wish to gather posts from. The link you wish to scrape can be gotten by going to instagram.com, typing the location you seek in the search bar, selecting the result that represents your location, and taking the url from that page. Your input should look something like this: https://www.instagram.com/explore/locations/238370172/orono-maine/
  <u>Twitter</u>
  If Twitter is selected, the textbox will take in a list of locations. These locations will be added to the search parameters so that only tweets taken with that location tag will be returned. Each location inputted should be preceded by the hashtag symbol (#). The input should look something like this: #newyork#boston#washington,D.C.
- Phrases Textbox:
  The Phrases Textbox is only available for Twitter searches and will not be visible if Instagram is selected. The textbox takes in a list of phrases that can be searched by. These phrases will be added to the search parameters, so that only tweets with that phrase inside it will be returned. Each phrase inputted should be preceded by the hashtag symbol (#). Phrase input should look something like this: #planking#I love bears#working late
- Multiple Twitter Inputs
  For best results, the logic behind twitter inputs should be known. If more than one of a type of input is used, they will be ORed together. This means that in the Locations Textbox, if #newyork#boston was inputted, tweets with either the location tag newyork or the location tag boston will be returned. It should also be known that if more than one type of input is used, they will be ANDed together.

This means that if the hashtag #dog and the phrase “I love lucy” were both inputted, only tweets containing the hashtag #dog and the phrase “I love lucy” would be returned. This also means that if both #dog and #cat were inputted, but the locations boston and newyork were also inputted, then you would have four types of tweets returned. Those being #dog + location:boston, #dog + location:newyork, #cat + location:boston, and #cat + location:newyork. Finally, no more than 9 total terms across hashtags, locations, and phrases should ever be used during one scrape.

- Multiple Instagram Inputs

  Instagram cannot take multiple inputs. You should only ever input one hashtag or one location link per search.
- Start Date Textbox:

  The Start Date Textbox is only available for Twitter searches and will not be visible if Instagram is selected. It takes in text information representing the farthest date in which tweets are accepted to be from. Combined with the End Date Textbox a date range for tweets can be created. The start date should not be any date later than 30 days from the current date. The start date should be inputted in the form of MM/DD/YYYY
- End Date Textbox:

  The End Date Textbox is only available for Twitter searches and will not be visible if Instagram is selected. It takes in text information representing the latest date in which tweets are accepted to be from. Combined with the Start Date Textbox a date range for tweets can be created. The end date should not be any date later than 30 days from the current date. The end date should be inputted in the form of MM/DD/YYYY
- Settings Button:

  Upon pushing this button the user will be navigated to the Settings Page.
- Home Button:

  Upon pushing this button, the user will be navigated to the Home Page.
- Continue Button:

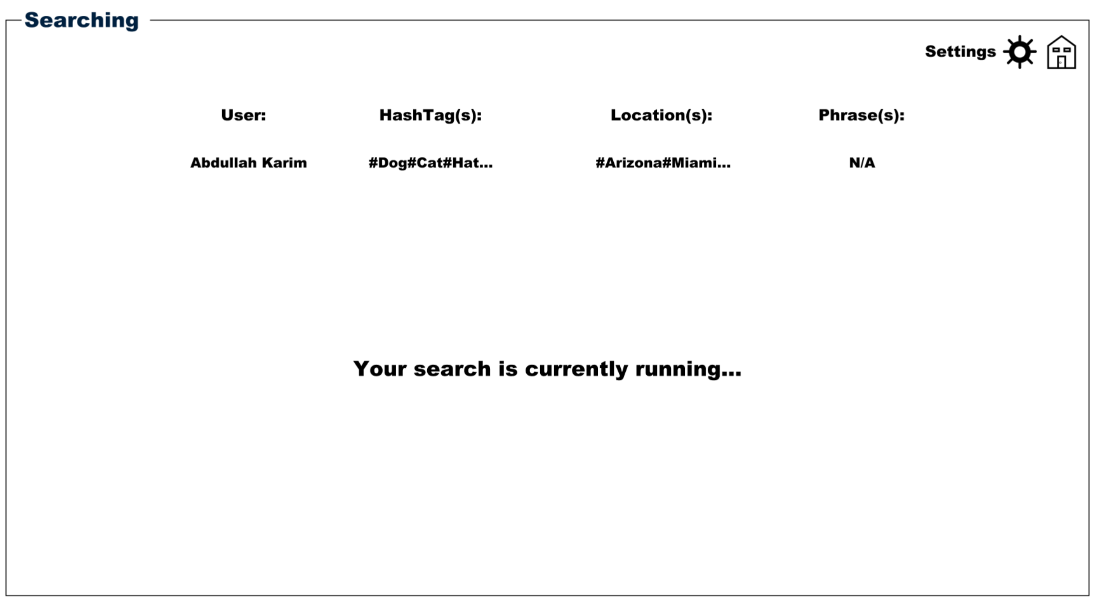
  Upon pushing this button, a new scrape will be started according to the specified requirements in the above text boxes. The user will then be navigated to the Searching Page where they can wait until they wish to stop their scrape.

Probable Errors and Possible Causes

- Invalid or wrong input

  May possibly cause errors with scraping
- There are multiple things that need to be tracked by the administrator for this system. They are all found in the Administrator Manual. Should you have valid input, but scraping still isn’t working, contact your administrator for help.

Searching Page



User:	HashTag(s):	Location(s):	Phrase(s):
Abdullah Karim	#Dog#Cat#Hat...	#Arizona#Miami...	N/A

Your search is currently running...

Functional Description of Page

The Searching Page is used to monitor your current search. With it you can view the searches parameters and the number of posts scraped so far. When a scrape finishes it will automatically download the user's scrape for them within their browser.

Cautions and Warnings

N/A

Procedures

The Searching Page has 2 buttons. They function as follows:

- Settings Button

    The Settings Button navigates the user to the settings page.

- Home Button

    The Home Button navigates the user to the home page.

Probable Errors and Possible Causes

N/A

A. Error Messages and Recovery Procedures

The user will not be presented with error messages. However, if the user opened their browser's console, they could potentially view error messages. If there are error messages, attempt to close the tool and reopen it in a new window. If the error messages persist, contact an administrator to resolve the issue. Closing the tool may lose any data that has not already been downloaded. This includes all scrapes that are currently in progress, resulting in the scrapes ending without any data being collected.

System Recovery Procedures can be found in the S.S. Media Administrator Manual.

B. GLOSSARY

Data

A piece of information such as an email address. The email address itself is data that can then be stored in a database.

Database

A database is a form of storage for information such as a person’s email address and password.

Scrape

The searching and collection of information from a website.

Appendix A - Team Review Sign-off

All members of Black Bear Analytics have gone through the review process for this document and agree on both the manual entries in this document and are certain there is enough coverage to be effective. We have also received a sign-off from the client, approving this document which means they have reviewed and have accepted what is listed above until fixed.

Name	**Signature**	**Date**
Customer:		
______________________	______________________	______________________
Comments:		
Team:		
Ryan M. Handlon	[illegible]	April 19th, 2021
Comments:		
Abdullah I. Karim	[illegible]	April 19th, 2021
Comments:		
Griffin L. Fluet	Griffin L. Fluet	April 19th, 2021
Comments:		
Colleen DeMaris	Colleen DeMaris	April 19th, 2021
Comments:		
James West	James West	April 19th, 2021

Appendix B - Document Contributions

Abdullah, Colleen, Griffin, James, and Ryan all contributed equally (20%) to this document. Each member added to a section(s) of the document with valuable content. In drafting this document each member listed defects they had, solutions for problems, and general information to help a user operate S.S. Media.