Product: S.S. Media

Client: Doctor Judith E. Rosenbaum

Administrator Manual

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 Rasberry 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 only 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 Putty (for Windows) should be installed. 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 using 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 python 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”}
```

<u>Backup Periodic Administration:</u>
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:

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

Comments:

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

Comments:

Griffin L. Fluet	Griffin L Fluet	April 7th, 2021

Comments:

Colleen DeMaris	Colleen DeMaris	April 7th, 2021

Comments:

James West	James West	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.