# Metaforge

by

Christopher Morris and Collin Mockbee

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology



| | |
|---|---|
| Christopher Morris | 04/15/2019 |
| Collin Mockbee | 04/15/2019 |
| Tyler Hopperton, Faculty Advisor | 4/15/2019 |

University of Cincinnati
College of
Education, Criminal Justice, and Human Services

April 2019

# Table of Contents



## List of Illustrations

## TABLES



## FIGURES

# Abstract

Information about data – known as Metadata, has the potential to provide a vast range of knowledge about data or a target in question. Metadata can provide information including the creation date, geolocation, software used to create the file and more – which can be useful for many users in the information security field. We created Metaforge to fill the gap in the information gathering arsenal to provide the user with a clean and efficient program. The application includes a simple GUI that creates a dynamic HTML report with multiple pages that displays raw and filtered metadata and a useful statistics page. Current existing metadata tools are outdated and unmaintained which results in a difficult user experience. The capabilities Metaforge includes are detailed metadata analysis, a practical report, and efficient functionality with the final outcome of a positive user experience.

# Introduction

## 1.1 Introduction

Metadata is described by the Merriam-Webster dictionary as “data that provides information about other data.”. Metadata can provide valuable information about many file types which can help when gathering information about an organization or institution. This information can be helpful for one performing a penetration test or audit. In general, gathering metadata is a type of information gathering technique which can be categorized as an OSINT (Open-Source Intelligence). We would like to provide a solution to metadata analysis. We propose to make an easy-to-use, fluid, effective, helpful and a well-documented OSINT tool that analyses metadata – made for an auditor or pentester which would be easy for anyone to use.

## 1.2 Problem

There are a few programs that do metadata analysis that are unmaintained, difficult-to-use and broken. Some examples are FOCA and Metagoofil. Metagoofil’s code on Github has last been updated six years ago and use deprecated and broken libraries. The issues tab in Github list many complaints that explain why the tool doesn’t work correctly and doesn’t accomplish its goal. Even when these tools manage to work, they don’t provide a larger picture of what the metadata can provide. For example, FOCA’s analysis of a picture did not provide the important metadata that we found with a standard metadata viewer, such as height, width and size of the file. There are many different types of tags that metadata provides that give valuable information about the source which these tools cannot effectively process in an organized manner.

## 1.3 Solution

Create a working, easy-to-use and useful python application that will scan files in a given directory or website that will output an HTML file. The HTML file will be in a report-style format that will designed to show the metadata holistically and will also show the raw data from the Exiftool. The HTML file will also have graphs that depict different metadata tags. Users will be able to gather information from sources of interest that may provide metadata that will be organized into profiles that fit the criteria of the specified source. Each project that the user creates has specific data to the source and will be compiled in an organized fashion to be easily retrievable.

## 1.4 Project Goals

Over the course of the two semesters, we have thought of certain goals for the Metaforge project. The goals included were to make an easy to use application, make a useful and good-looking report, create an efficient program flow and to save time for the user when doing metadata analysis. In all, the program should make metadata analysis easy, efficient, and useful for the user.

## 1.5 Overview

In all, the introduction section gives a condensed version of what will be discussed in the later sections of this paper. The rest of the report will depict the project concept, design objectives, methodology, user profile, technical architecture and discussion, testing, budget, WBS and Gantt Chart, problems encountered, future recommendations and the conclusion.

# Discussion

## 2.1 Project Concept

Effective design and technical elements are vital for any application. We have chosen and designed our application very carefully to make sure that the user experience is the best. The core design of our project is Figure 1. The figure shows the flow of the user's interaction with the application.

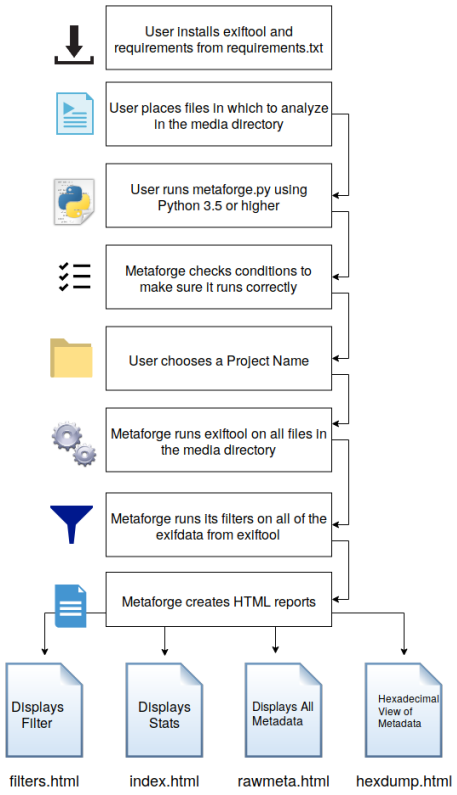


**Figure 1: Metaforge's Design**

### 2.1.a Filter

The Filter is another vital part of the design of the application. The filter will allow the user to view only the important tags & information about the metadata. The filter will come in handy if the user inputs hundreds or even thousands of files to view for metadata analysis. Instead of looking through thousands of lines of mostly useless information, the user will only have to look through a hundred, thanks to the filter. The filter's design is depicted in Figure 2. The filter will essentially remove unimportant while leaving the important.

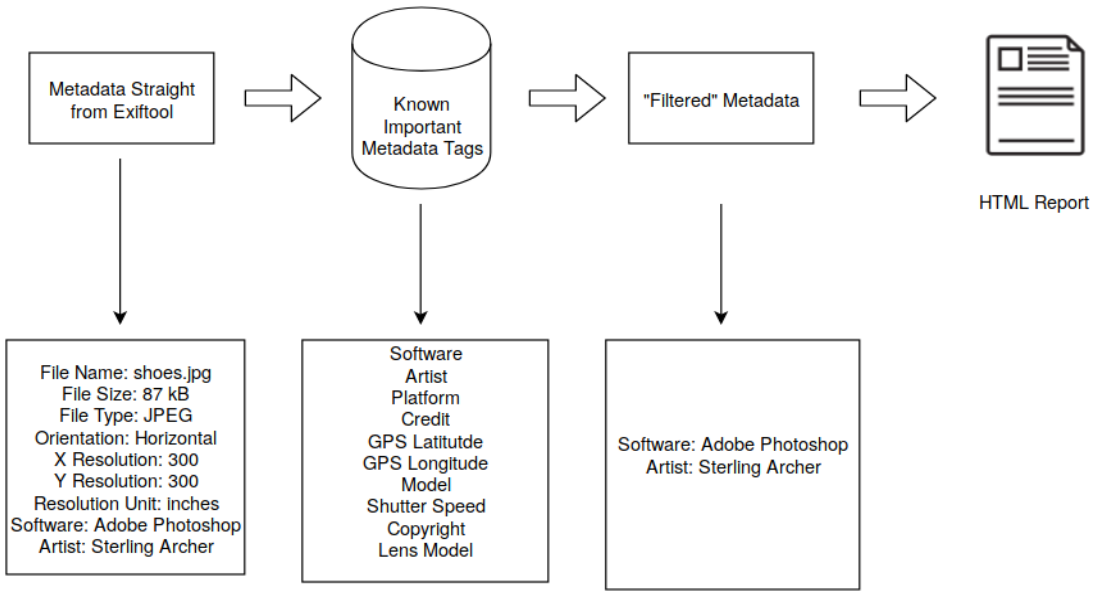


**Figure 2: Filter Design**

### 2.1.b Report

The report will be the aesthetically pleasing part of the application. The report will detail the raw metadata (without filter), the filtered metadata, and the hexadecimal values of all the metadata. In other metadata analysis program that we have looked at, their reports were not good-looking and did not contain a lot of information. Our goal of the reports is to make them pretty, useful, and detailed. The report will be filled with visually pleasing CSS and JavaScript.

## 2.2 Design Objectives

Our goal was to create a working, easy-to-use and useful python application that will scan files in a given directory or website that will output an HTML file. The HTML file will be in a report-style format that will designed to show the metadata holistically and will also show the raw data from the exiftool. The HTML file will also have graphs that depict different metadata tags. Users will be able to gather information from sources of interest that may provide metadata that will be organized into profiles that fit the criteria of the specified source. Each project that the user creates has specific data to the source and will be compiled in an organized fashion to be easily retrievable. Below is Table 1 that has our design objectives that we had created in the early stages of Metaforge.

| | |
|---|---|
| HTML Report (High) | An easy to read and navigable report is essential for the user to view the contents of the metadata analysis. |
| Organization of Metadata (High) | The user will want to easily group together similarities and patterns found in the metadata. |

| | |
|---|---|
| Documentation (High) | Informational documents will provide the essential steps to effectively use the tool and retrieve the best results. |
| Python 3.5 (Medium) | It is not essential for the program to be in Python, but the language is perfect for our scenario. |
| Easy-to-use UI (Medium) | Our target audience is primarily for those in IT, so they will be able to use their previous knowledge to understand how it works. |

**Table 1: Design Objectives**

One example of an idea that we ended up abandoning was including a Web Crawler that had the potential to gather information from a target URL. We thought that this would impractical for the idea that we had for our project. We kept the scanning process where all the files in question would be scanned from a local directory located within the tool itself. If given more time to work on the project, we may have added a feature that allowed for this kind of metadata to be processed.

## 2.3 Methodology

### 2.3.a Design Requirements

In order for the Metaforge project to be successful, we laid out a couple of design requirements. For example, the application must run on Python 3.5 or higher. We want the code to be the most current version. We also want to make Metaforge for Linux/Unix machines. Many information security professionals and researchers use Linux (our target audience) so it's

definitely a design requirement. For the reports, we want them to be interactive and good-looking, so we thought that HTML/CSS3 reports would be a fine choice. Finally, for the best user experience, we want to include documentation, so the user can easily navigate the application.

### 2.3.b Procedures

To keep us on track with the successful completion of Metaforge, we had numerous procedures that we kept up on. Our constant research involving metadata and various tools that could process metadata helped give an idea as to what kind of objectives we set in the first place. While we worked on the features of our project, we also went through the process of learning Python 3 and familiarizing ourselves with the libraries. As a team, we came together every week to discuss the features of Metaforge, testing and discuss the final bug fixes for the final product.

## 2.4 User Profile

**Potential Users:**

Information security professionals, law enforcement, auditors, and technologically adept hobbyists

**Software, Interface and Related Experience:**

All users will be running the application with a command line interface.

**Experience with Similar Applications:**

The Metadata Analysis project was made for technologically adept users that know how to use a command line and know how to run a python script. The user will most likely use the tool for information gathering.

-FOCA

-Metagoofil

-Bulk-Extractor

**Task Experience:**

From the very beginning of starting the python application, the user will be prompted to create a project name and define a directory of the media files they wish to analyze. After inputting those into the application, the user will give all of the raw metadata of the files, a hexadecimal view of the files, and a shortened, more useful report that will show only the important metadata tags. The report will be in html and can be viewed with ease.

**Frequency of Use:**

The software itself will be used by IT professionals who are interested in gathering large amounts of metadata quickly and efficiently. The tool is intended to be able to handle large amounts of data, meaning the tool would be used for multiple tasks at once. The additional features planned down the road may entice users who plan to use the tool more frequently in gathering metadata for fields in IT that deal with cybersecurity and data forensics.

**Key Interface Requirements that the Profile suggests:**

-Visually pleasing, efficient, and easy-to-read report

-Potential intuitive UI

-A filter to display only the major and important metadata tags

-Relatively fast processing of the metadata

-Documentation to aid in using the tool

## 2.5 Use Case Diagram

Figure 3 is the Use Case Diagram. This describes the requirements and the technology which will be used to create the project. It also defines relationship between requirements and technology.

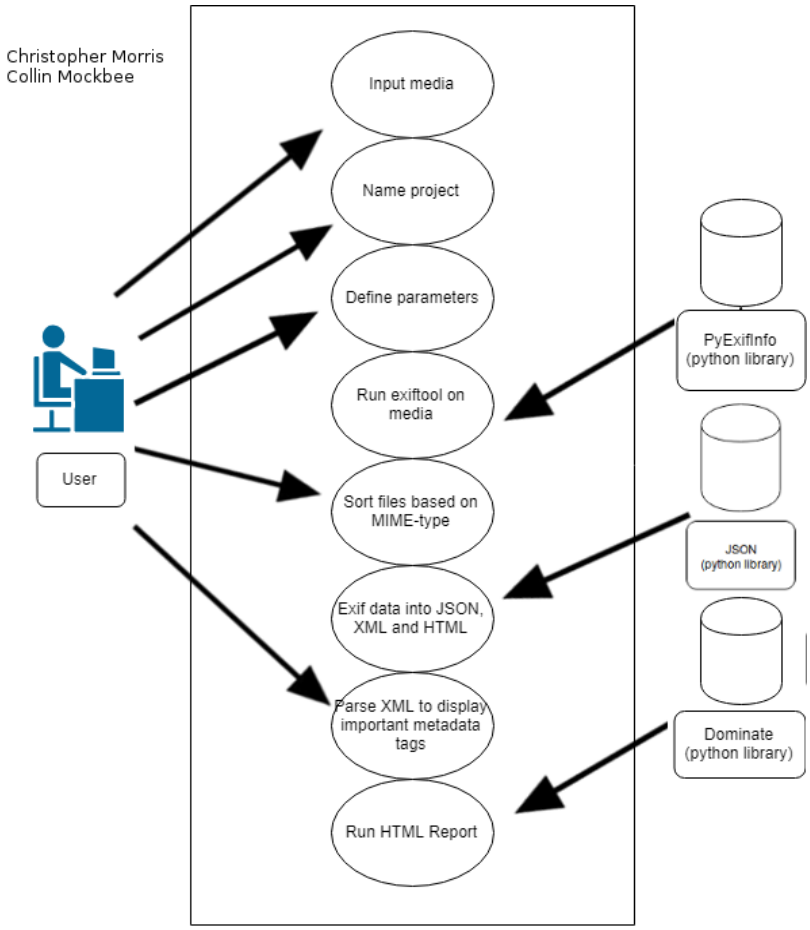


**Figure 3: Use Case Diagram**

## 2.6 Technical Architecture

Figure 4 includes the software architecture for Metaforge. Metaforge.py is the main python class which calls the rest of the python classes. The python classes are markups.py, exiftool.py, fileinteractions.py, and filter.py. Definitions.py defines the root directory of the program, so it is placed at the root directory of the program in order to define it.

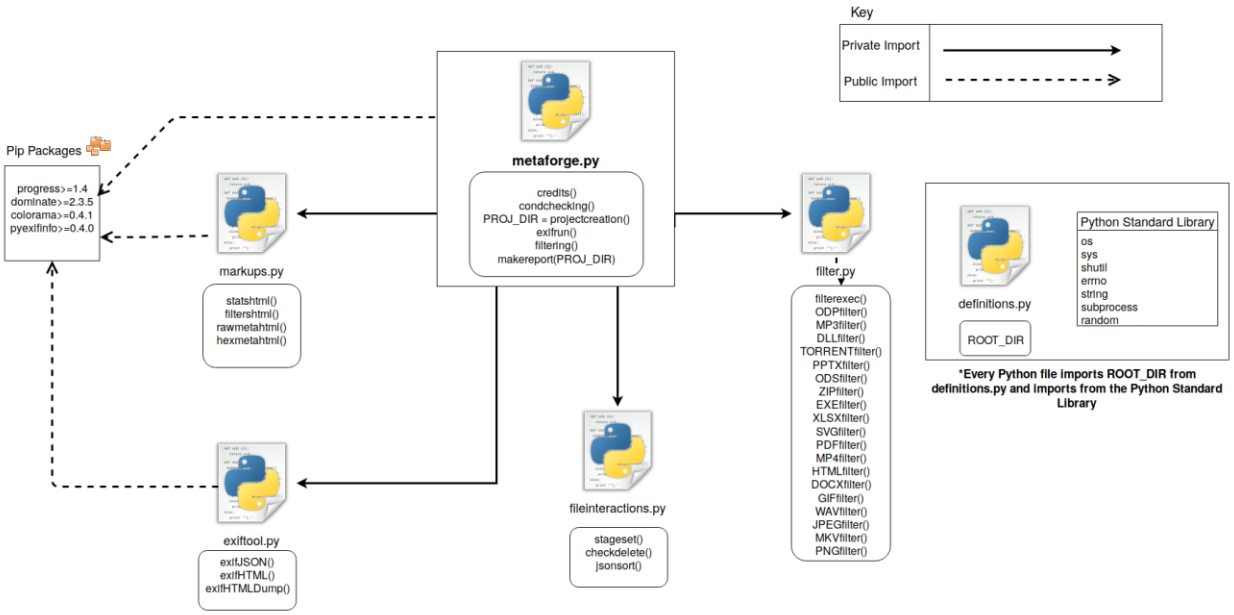


**Figure 4: Software Architecture**

## 2.7 Technical Discussion

Each file in the Metaforge project serves an important role. The main 'control center' of the application is metaforge.py. The file calls all the functions that are located in the other 5 python files. It is the blueprint of the project and is like an instruction manual. The first step that metaforge.py takes is to check to make sure that the conditions are correct. The file that does this is fileinteractions.py. It makes sure that the directory structures are in line and present. After it

completes checking, metaforge.py executes exiftool.py which runs the exiftool on all of the files defined in the media directory. When the exiftool is done running, metaforge.py executes filter.py, which filters out useless metadata in the dataset. The next file up for discussion is markups.py. This file lays out and generates the html report. All the stats are calculated and processed here. The conclude the program, metaforge.py informs the user of where the final report is and safely exits.

## 2.8 Testing

### Testing Overview

The testing methodology for the Metadata Analyzer application will improve the overall installation, functionality, and user-experience of the application. We chose this approach because these three operations will be key the the overall quality and practicality of the application. Individuals who wish to use this section includes:

- Developers
- Project Managers
- Members of the Team
- Q&A Testers

**Scope**

The scope of the testing is to test the installation, functionality and the user's experience of the application. The tests will be organized in three different parts to fulfill the requirements of the application. The installation portion of the testing will involve multiple variants of Unix-based Operating Systems.

**Objectives**

The objective of testing is to verify that the application is easy to install and setup with the least amount of complications. The testing phases will be setup to gauge the usefulness of the application as well as the ease-of-use.

**Logging Test and Procedures**

The information that is gathered from the testing will be documented and assessed to better understand what problems the application has and how these problems can be resolved. The documented bugs will be given priority listing to organize the process of improving the application to its fullest potential.

In our documentation for testing, we plan on including:

- Tester Details

- Date
- OS used to test
- Ease of Installation (1 difficult - 10 easy) + comments
- Ease of Use rating (1 difficult-10 easy) + comments
- Practicality (1 difficult - 10 easy) + comments
- Outcome (pass/fail)

**Testing Procedures**

1. Installation Setup - This test will demonstrate the ease of meeting the requirements to download the application correctly. The test will verify that Unix-based Operating Systems can run the application correctly.
2. Functionality Test - This test will verify that the application runs correctly on different platforms and configurations. This is where most of the information and bugs will be gathered from testing.
3. User-experience Test - This test will give us information in how the user reads the reports, what information they find valuable, and the usability of the application.

**What We Learned**

Throughout the testing phases of our application, we have gathered information that has proved to be very useful for future planning and future tests. The testers that were involved in the process were helpful in providing feedback and critiquing the functionality of the application in a way that was positive for the overall success of the testing phases. With the new information, we

plan on implementing and upgrading features such as, better error handling and documentation for settings up pip3 and python3 for those who have never used it before or for those who are using outdated versions. Below in Table 2 are the results from the users that we asked to test Metaforge.

| # | **Tester** | **Date** | **OS used** | **Ease of Installation *(1-10) + comments*** | **Ease of Use *(1-10) + comments*** | **Practicality *(1-10) + comments*** | **Outcome *(pass/fail)*** |
|---|---|---|---|---|---|---|---|
| 1 | IT Student | 02/10/2019 | CentOS 7 | 4: had a hard time pip packages b/c exiftool needs to be installed first | 3: no issue at all - except maybe require user to run "chmod +x run.py" | 3: easy to use, maybe handle errors better | Fail, See notes below |
| 2 | IT Student | 02/11/2019 | Ubuntu 18.04 | 2: maybe create a pip3 installation section | 1: no comments, worked well | 1: no comments, worked fine | Pass, worked well |
| 3 | Non-IT Student | 2/11/2019 | MacOS 10.14.3 | | | | |
| 4 | IT Student (2nd attempt) | 2/13/2019 | CentOS 7 | 1: no comments, just had to run python36 metaforge.py | 1: no comments, just had to run python36 metaforge.py | 4: index.html didn't render, later fixed | Pass, See notes below<br><br>Found issue: subprocess.run doesn't work in python versions before 3.5! |
| 5 | Tyler H. | 2/18/19 | Ubuntu 18.04 | 1 - no comment Bad keyboard | 1 - no comments | 8 - graphical andstats of all the analyzing of everything | Pass |

**Table 2: Testing Results**

**Test User 1:**

Bug 1 (minor):

-When running exiftool to hexadecimal - there's an error

*"Warning: Install Archive::Zip to decode compressed ZIP information"*

Bug 2 (major)

-When trying to run Step 5, the statshtml() does not run at all and gets this error

*ERROR: could not run stat html report <class 'AttributeError'>*

Bug 3 (medium):

When trying to run Step 5, the html files do not get copied to the User's directory.

*ERROR: could not place reports into project directory <class 'FileNotFoundError'>*

This most likely failed because there was no index.html to be moved.

**Test User 2:**

No bugs, had an annoying time setting up pip3 and installing from requirements.txt, maybe add help to the documentation section.

**Test User 3:**

Only real issue was the report squished most of the text in the “Number of Files in each Filetype” of the index.html report.

https://wsvincent.com/install-python3-mac/

-xcode (Apple's xcode package)

-homebrew (package manager)

**Test User 4**

Bug 1 (major): index.html still not rendering

*ERROR: Failed to copy index.html-- <class 'FileNotFoundError'>*

*ERROR: could not run stat html report <class 'FileNotFoundError'>*

Also, narrowed down the errors:

-Filetypes chart works,

-Filesize chart doesn’t work

*Error: Filesize chart generation failed: (<class 'AttributeError'>, AttributeError("'module' object has no attribute 'run'",), <traceback object at 0x7f9e9037f708>)*

-Filtered vs Raw chart doesn’t work

*Error: Filtered vs Raw chart generation failed: (<class 'AttributeError'>, AttributeError("'module' object has no attribute 'run'",), <traceback object at 0x7f9e9037f708>)*

Found issue: subprocess.run doesn't work in python versions before 3.5!

## 2.9 Budget

Metaforge has not real infrastructure cost, however, we spent about 155 hours on various different parts of our project. These different parts include: planning, research, writing the code itself, the Senior Design Assignments and building the GitHub Pages site including the documentation that was with that. Our plan was to keep our project open-source and it to be a community tool that anyone could access and help to improve after the final stages.

## 2.10 WBS and Gantt Chart

*Table 3 depicts the WBS/Schedule that our team followed in order to keep organized and informed about important deadlines. One can note that on that right side it depicts the time we must complete the assignment and well as the description in the center.*

| Metadata Project WBS | | | |
|---|---|---|---|
| **Start Date** | **End Date** | **Desciption** | **Duration (Days)** |
| 8/26/2018 | 09/03/2018 | Team Members and Project Name (Assignment 0) | 7 |
| 09/03/2018 | 09/24/2018 | Team Contract (Assignment 1) | 21 |
| 9/24/2018 | 10/15/2018 | Project Abstract (Assignment 2) | 21 |
| 9/24/2018 | 10/15/2018 | Team Contract Resubmission (Assignment 3) | 21 |
| 10/1/2018 | 12/31/2018 | Practice and Research Python 3.5 | 91 |
| 10/15/2018 | 10/22/2018 | User Profile Submission (Assignment 4) | 7 |
| 10/15/2018 | 10/22/2018 | Use Case Diagram (Assignment 5) | 7 |
| 10/10/2018 | 10/29/2019 | 3-Minute Elevator Speech | 19 |
| 10/22/2018 | 11/5/2018 | Draft Report (Assignment 6) | 21 |
| 11/5/2018 | 11/26/2018 | Final Report (Assignment 7) | 21 |
| 11/26/2018 | 12/3/2018 | Material for Oral Presentation | 7 |
| 11/26/2018 | 12/3/2018 | Oral Presentation | 7 |
| 1/14/2019 | 1/21/2019 | Start requirements for Application (Libraries) | 7 |
| 1/21/2019 | 1/28/2019 | Start to think of requirements for web crawler | 7 |
| 1/28/2019 | 2/4/2019 | Design the HTML template (Draft 1) | 7 |
| 2/4/2019 | 2/11/2019 | Documentation - Draft 1 | 7 |
| 2/4/2019 | 2/11/2019 | Team Contract - Review p1 | 7 |
| 2/11/2019 | 2/18/2019 | Start drafting of the architecture of Python code | 7 |
| 2/18/2019 | 2/25/2019 | Version 1 - Complete | 7 |
| 2/25/2019 | 3/4/2019 | Testing of application | 7 |
| 3/4/2019 | 3/11/2019 | Review Comments and Start Code Revisions | 7 |
| 3/11/2019 | 3/18/2019 | Design the HTML template (Draft 2) | 7 |
| 3/18/2019 | 3/25/2019 | Documentation - Draft 2 | 7 |
| 3/18/2019 | 3/25/2019 | Team Contract - Draft 2 | 7 |
| 3/25/2019 | 4/1/2019 | Finish Architecture of the Python Code | 7 |
| 3/25/2019 | 4/1/2019 | Version 2 - Complete | 7 |
| 3/25/2019 | 4/1/2019 | Documentation - Final | 7 |
| 3/25/2019 | 4/1/2019 | Team Contract - Final | 7 |
| 3/25/2019 | 4/1/2019 | Convert Documentation to Latex | 7 |
| 3/25/2019 | 4/1/2019 | Convert Team Contract to Latex | 7 |
| 01/14/2019 | 04/09/2019 | IT Expo | 1 |

**Table 3: WBS Schedule**

*The project schedule below, Table 4, shows our Gantt Chart including each assignment or deliverable along the way. We measured our progress in months to easily visualize the status of our expected completion, per assignment.*

| TASK TITLE | START DATE | DUE DATE | DURATION |
|---|---|---|---|
| Team Members and Project Name (Assignment 0) | 8/26/2018 | 09/03/2018 | 7 |
| Team Contract (Assignment 1) | 09/03/2018 | 09/24/2018 | 21 |
| Project Abstract (Assignment 2) | 9/24/2018 | 10/15/2018 | 21 |
| Team Contract Resubmission (Assignment 3) | 9/24/2018 | 10/15/2018 | 21 |
| Practice and Research Python 3.5 | 10/1/2018 | 12/31/2018 | 91 |
| User Profile Submission (Assignment 4) | 10/15/2018 | 10/22/2018 | 7 |
| Use Case Diagram (Assignment 5) | 10/15/2018 | 10/22/2018 | 7 |
| 3-Minute Elevator Speech | 10/10/2018 | 10/29/2019 | 19 |
| Draft Report (Assignment 6) | 10/22/2018 | 11/5/2018 | 21 |
| Final Report (Assignment 7) | 11/5/2018 | 11/26/2018 | 21 |
| Material for Oral Presentation | 11/26/2018 | 12/3/2018 | 7 |
| Oral Presentation | 11/26/2018 | 12/3/2018 | 7 |
| Start requirements for Application (Libraries) | 1/14/2019 | 1/21/2019 | 7 |
| Start to think of requirements for web crawler | 1/21/2019 | 1/28/2019 | 7 |
| Design the HTML template (Draft 1) | 1/28/2019 | 2/4/2019 | 7 |
| Documentation - Draft 1 | 2/4/2019 | 2/11/2019 | 7 |
| Team Contract - Review p1 | 2/4/2019 | 2/11/2019 | 7 |
| Start drafting of the architecture of Python code | 2/11/2019 | 2/18/2019 | 7 |
| Version 1 - Complete | 2/18/2019 | 2/25/2019 | 7 |
| Testing of application | 2/25/2019 | 3/4/2019 | 7 |
| Review Comments and Start Code Revisions | 3/4/2019 | 3/11/2019 | 7 |
| Design the HTML template (Draft 2) | 3/11/2019 | 3/18/2019 | 7 |
| Documentation - Draft 2 | 3/18/2019 | 3/25/2019 | 7 |
| Team Contract - Draft 2 | 3/18/2019 | 3/25/2019 | 7 |
| Finish Architecture of the Python Code | 3/25/2019 | 4/1/2019 | 7 |
| Version 2 - Complete | 3/25/2019 | 4/1/2019 | 7 |
| Documentation - Final | 3/25/2019 | 4/1/2019 | 7 |
| Team Contract - Final | 3/25/2019 | 4/1/2019 | 7 |
| Convert Documentation to Latex | 3/25/2019 | 4/1/2019 | 7 |
| Convert Team Contract to Latex | 3/25/2019 | 4/1/2019 | 7 |
| IT Expo | 04/09/2019 | 04/09/2019 | 1 |

**Table 4: Gantt Chart**

## 2.11 Problems Encountered

Over the course of Fall and Spring Semester, we ran into some problems which must be solved in order to create a successful Senior Design Project. For example, some issues included trying to utilize outdated libraries and learning Python 3.5. Outdated libraries are an issue because it is the sole reason why older programs didn't work and Metagoofil is a great example. To try and avoid this, we will attempt to use well-maintained libraries which people use and patch. Another issue we had was learning Python 3.5. Python 2, the older version of Python will be deprecated next year. Python3.5's syntax is different than Python 2 so we had to learn it so our project would last longer. In all, we saw that other Metadata viewers were old and unmaintained. We came up with the solution to that problem by using newer libraries and the

newer version of Python. In addition, we will place the project on Github so people can edit and view the code for future improvements.

## 2.12 Future Recommendations

We currently feel that there are ways of making our report system look cleaner than its current state; to include a preview of each file that Metaforge processes within the report to help the user in locating information per file. Another feature that was talked about was adding the ability to create a PDF version of the report. The filter also has been a subject that could be improved with better design. Each filetype's metadata tag is defined in one tuple variable. The variable is very long could be re-created to be shorter and more efficient. Windows 10 support could also be a feasible feature for Metaforge.

# Conclusions

## 3.1 Fall Conclusion

To summarize the experiences, we had during the project and the skills we have developed, it was learning process that took time, communication, organization and constant awareness. Some of the main lessons we, as a group, took from this project were improved communication skills, project management and how to deal with project management. Some of the most important skills we found most notable are learning about the agile software development method, a better understanding of metadata as a whole and what it is capable of, improved knowledge in python syntax, and learning about LaTeX and what it can offer. As for the Spring of 2019, we plan on working on the project in tandem with our schedule that we created. Our goal is to provide a tool that is functional, unlike the counterparts that are currently available. We are creating our tool completely from scratch and creating our own code. This allows us to plan for every piece of functionality and helps us become more familiar with the capabilities of our tool as we learn more about the process of data manipulation and the organization of reports. The process of handling the assignments and constantly doing research on the topic of metadata, the python language and the libraries available to us and how to use them has shown us what lies ahead in creating our finished project to ensure its uniqueness and efficiency in the endgame.

## 3.2 Lessons Learned

Whilst working on our project as a team, we learned valuable lessons like working on a team, utilizing our resources (advisors and professors included), individual brainstorming/development, as well as what it takes to be a leader in certain aspects of our

project development plans. We were able to take away from these lessons what it takes to successfully work as a team and benefit the entire team as a whole. We worked hard to make sure that, as a team, we both understood the plans of the project and what was expected of the end-product.

### 3.3 Skills Learned

During the Fall and Spring semester of Senior Design, we learned many new technical and interpersonal communication skills. First off, we learned Python, which was huge deal because we did not know it prior to this course. These skills will be useful in the future when we are scripting or automating a task. We also learned about working on a team. We have worked on teams before, but this was different because it was for an IT project and it was only us two. We learned how to share ideas and work together.

### 3.4 Completed Since Fall

Since the fall, we have been able to: complete the code that the program requires, complete the HTML report system for where our information is displayed and create and GitHub Pages site for documentation pertaining to the project. We have added every feature that was included in our initial design plan and have created a fool-proof program that is capable of handling more files than we thought was capable. After the initial program was finished, we also added the feature to add your own custom tags to the filter allowing for customizable results and added functionality.

### 3.5 What we learned from Expo

During the expo, we had the opportunity to meet with lots of people with varying experience with metadata. We were given a lot of feedback on our project and what kind of purposes metadata might be useful for their everyday lives. We had a lot of really good conversation with the high school students that were curious about metadata and what kind of uses it had and had the pleasure of explaining the usefulness of Metaforge. During our judging panel, we were given really good feedback in terms of licensed software that companies used for metadata analysis and the differences between our open-source program and programs an enterprise might use. We even had a lot of groups get recommended to come to our booth specifically because of our interest in our topic and information we had to offer (so we were told).

# References

*"Metadata." Merriam-Webster, Merriam-Webster, www.merriam-webster.com/dictionary/metadata.*

Harvey, Phil. "Exiftool." *Https://Www.sno.phy.queensu.ca/~Phil/Exiftool/*, www.sno.phy.queensu.ca/~phil/exiftool/.

Martorella, Christian, et al. "Metagoofil." *Edge Security*, www.edge-security.com/metagoofil.php

Garfinkel, Simson L. "Bulk Extractor." *Https://Github.com/Simsong/bulk_extractor*, github.com/simsong/bulk_extractor.

Press, NISO. "Understanding Metadata." *Niso.org*, NISO Press, 2004, web.archive.org/web/20141107022958/http://www.niso.org/publications/press/UnderstandingMetadata.pdf.

"FOCA." *Eleven Paths*, Eleven Paths, www.elevenpaths.com/labstools/foca/index.html.

"Build Software Better, Together." *GitHub*, www.github.com/.

# Appendices



## Appendix A

### A.1

Metaforge's code is hosted on Github.com and is open for everyone to view. We chose Github as it is a social-code hosting platform for open-source projects. A.2 depicts the links to our Github project.

### A.2

The following are the links to our project's code and documentation.

Code

*github.com/chriswmorris/Metaforge*

Documentation

*chriswmorris.github.io/Metaforge/*

# Appendix B

## B.1
More on Metadata

Metadata is a very broad term and can be applied to many situations. A great way to learn more about Metadata is to read Understanding Metadata by The National Institute of Standards Organization (NISO). To read the document, please visit their website, linked below.

*web.archive.org/web/20141107022958/http://www.niso.org/publications/press/UnderstandingMetadata.pdf*

# Appendix C

## C.1
**Project Poster** (Figure 5)

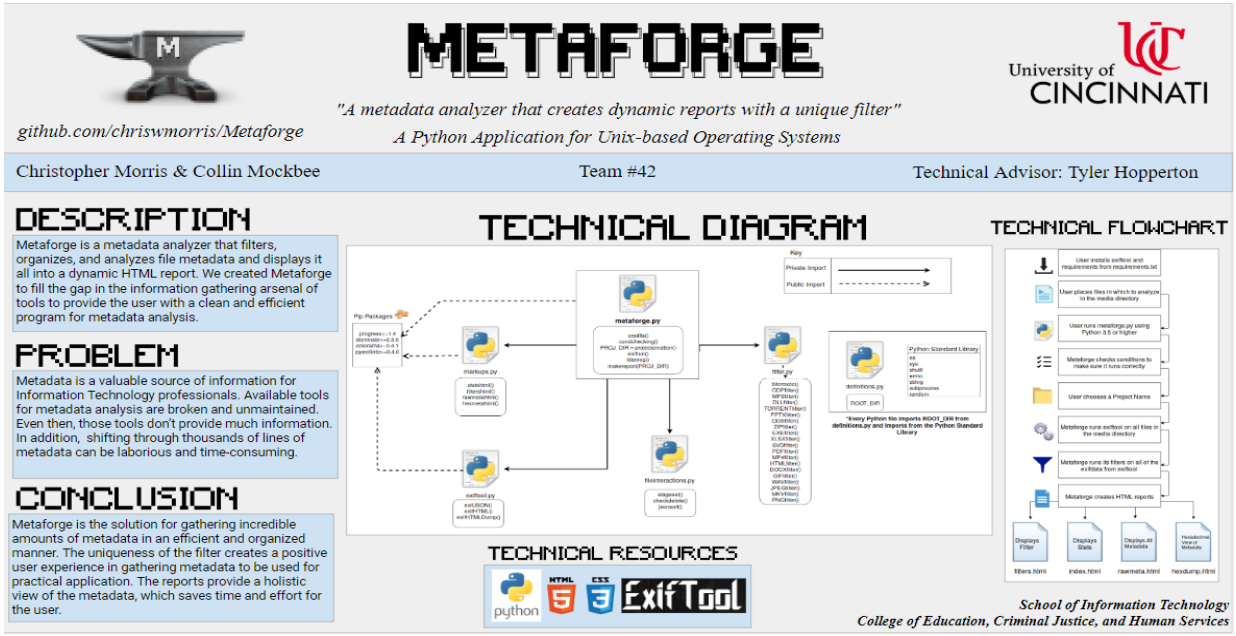


**Figure 5: Project Poster**