CSC3600 ICT Professional Project

File Metadata Harvester and Searcher

Project Plan

Version No: 5

Document ID: 1.3

Aug 08, 2019

**Team Name**: *USQ Learning Emporium*

**Team Members:**

Jesse Hare W0070949

James MacKeown U1093772

Ryan Sharp U1090072

Richard Dobson U1011744

Vincent Roberts U1046454

Supervisor: Stijn Dekeyser

Project Plan Version: 4

Version History

|  |  |  |
| --- | --- | --- |
| **Version** | **Change** | **Date** |
| 1 | Created Doc Template | 2/08/2019 |
| 2 | Added Content to Introduction | 6/08/2019 |
| 3 | Added content to the Specification | 9/08/2019 |
| 4 | Added more content to parts 1 and 2 | 12/08/2019 |
| 5 | Addition to Risk management plan | 13/08/2019 |
|  |  |  |
|  |  |  |

Table of Contents

[1 Introduction 3](#_Toc16693337)

[The Project 3](#_Toc16693338)

[Motivation 3](#_Toc16693339)

[Ultimate Goal 4](#_Toc16693340)

[General plan 4](#_Toc16693341)

[2 Project Specification 5](#_Toc16693342)

[Project details 5](#_Toc16693343)

[Component 1: File Metadata Harvester 5](#_Toc16693344)

[Component 2: File Metadata Searcher 7](#_Toc16693345)

[Problems (Scope, Limitations, Assumptions) 8](#_Toc16693346)

[Objectives 10](#_Toc16693347)

[Expected Outcome and deliverables 10](#_Toc16693348)

[3 Project Design 11](#_Toc16693349)

[4 Work Break-down Structure and Task Scheduling 11](#_Toc16693350)

[5 Time and Cost Estimation 11](#_Toc16693351)

[6 Project Schedule 11](#_Toc16693352)

[7 Risk Management Plan 12](#_Toc16693353)

[8 Code of Conduct 13](#_Toc16693354)

[9 Conclusions 14](#_Toc16693355)

[References 14](#_Toc16693356)

[Appendix 14](#_Toc16693357)

[Contribution Statement 14](#_Toc16693358)

# 1 Introduction

## The Project

The purpose of this project our team will be undertaking is to provide a tool for our client to harvest metadata for files, and another tool that will use that harvested data and display it in a user-friendly GUI.

The two components of this project are:

1. A script that is invoked via the CLI (command line interface) that takes a directory path as input and returns (on Standard Output) CSV-formatted text having one line per file in the directory and listing all metadata attributes and values extracted from that file.
2. Develop a GUI that reads the output of the CLI script (CSV file) and display a spreadsheet-like grid, that can be interacted with. Examples of interactions include sorting by specified attributes, searching for file information on multiple criteria and double clicking a file opens it.

The project is to be implemented in Python, and the desired platform that the components of the project are to run on is Linux, although depending on time constraints we may expand support to other OSes. (Such as Windows or MacOS).

The result of this project will be a pair of tools that will fulfil all our client’s needs/requirements. Due to time constraints we will need to be careful that our project remains within a well defined scope, which we will do so by carefully understanding what is it our client wants from the product vs what will be achievable in the timeline we have been presented with.

## Motivation

Following an initial meeting with our client, we have come to understand the motivation behind our client requesting our team to undergo this project.

The client wishes to be able to harvest file metadata and then be able to interact with that data. The data will be represented in a GUI, which will list all files and their attributes in a spreadsheet-like manner. The client wants to be able to perform searching, sorting and file access operations on the file records represented in the GUI.

Our client will be able to efficiently harvest the metadata of files and display them in an interactive GUI that allow further sorting and searching by specific keywords/attributes, with each represented file having the ability to be opened from that same interface. This will offer another level of utility for our client that is unprecedented in most standard Linux distributions.

We also believe that for any OS file-management is an important task that needs to be performed regularly to ensure that the file-system remains organised and easy to navigate. We believe that this project will result in a delivered product that helps our client perform this duty.

## Ultimate Goal

The suite of functionality requested by our client closely resembles those offered by several existing file-managing products. The difference between our proposed product and existing software is that the tools we are developing will allow our client to view metadata that is usually not displayed by typical file manager software, or readily accessible. This will allow our client to view file information in more detail, and to search for file information in more specific parts of the filesystem, and against more specific criteria.

We expect this tool to be invaluable in searching for specific files, whose location may no longer be known, and perhaps even the filename itself may have been forgotten. Misplaced files or files that are difficult to find should be found easily enough if the general location of the file is at least known.

But most importantly of all, our goal is to satisfy all user requirements and expectations. We hope they find our software solution as useful as we believe it is going to be, and it serves them well for some time into the future.

## General plan

We will be using an Agile approach to how we are managing the project, utilising the Scrum framework to ensure that our team work efficiently and learns together to produce the final product. The general plan of how we will run the project is below.

1. Planning

* Establish the general objectives of the project
* Design the software architecture

1. Sprint Cycle (Repeat until project closure)
2. Assessment

* Review objectives
* Close involvement of client to introduce requirements/tasks
* Assign priorities and risk to product backlog/task list

1. Selection

* Team and customer work together to select the features and functionality to develop during the sprint

1. Development

* Develop the selected features and functionality agreed upon by the team and the client

1. Review

* Review the work done and present to the client
* If product incomplete, begin a new Sprint cycle
* If product is complete, and client is satisfied, proceed to next phase

1. Project closure

* Wrap up the project
* Complete documentation
* Reflect on learning experiences during the project

# 2 Project Specification

## Project details

The file metadata harvester and searcher requested by the client consists of two parts, the metadata harvester, and the searcher. The harvester program searches the operating system and creates the data that is then used as standard input into the metadata searcher. We will now delve deeper into the technical and non-technical details for this project.

### Component 1: File Metadata Harvester

The Metadata harvester is a script that is to be executed via the Command Line Interface. It takes as input a directory path, and returns via standard output CSV-formatted text, with one line per file in the directory, listing all metadata attributed and values available for each file.

The Harvester Script should have an option to search recursively – that is the user should have the choice of only searching the folder specified in the file path input, or also all subdirectories below the specified directory as well. An example of this tool being invoked via the CLI is below:



The sample above is an example of a CLI call to the harvester tool to recursively harvest all metadata in the Downloads folder, and output to a csv formatted file.

An abbreviated example of what the contents of the csv file generated would look like is as follows:

/home/Bob/Downloads/readme.txt,size:400,createdate:2018-07- 27,owner:Bob

/home/Bob/Downloads/vids/sample.avi,size:123000,length:1'32",dimensions:240x120

/home/Bob/Downloads/songs/jeremy.avi,size:321000,length:3'32",title:Jeremy,artist:Pearl Jam,rating:4

/home/Bob/Downloads/pics/20180727102501.jpg,size:89000,dimensions:640x240,camera:nikon,tags:wedding

The sample output above is based on harvesting file metadata in a directory and its subdirectories, with 4 files in total found.

#### Metadata Harvester Technical details

##### Language

The harvesting script will be written in Python, and any libraries included in the script will either be standard with Python or open-source.

##### Tools

The tools required to develop the script are:

* Computer running a Linux-based OS
* A text editor such as Atom, Vim, Sublime Text etc
* Latest version of Python will be used, for which an interpreter usually comes preinstalled on most popular Linux distributions

Most members of the team will most likely be using an IDE of some sorts, like PyCharm, Spyder etc.

##### Storage

The script itself, and any generated output files can be expected to take up a negligible amount of storage space on a typical system. The script itself shouldn’t be anymore than 100KB, and the generated CSV file should be quite small as well, even when a large target area with many files is harvested, it would be surprising for the script to generate output that would take up any significant amount of space on the host filesystem.

##### Software components

* Linux-based OS will be used for development and testing of the harvesting script and all required tools for development. It is most importantly required to run the script, as it will be designed to work with the Linux filesystems specifically.
* Latest version of Python will be used, for which an interpreter usually comes preinstalled on most popular Linux distributions
* A command line interface will be required to invoke the script, such as Bash. It would be unusual however for an OS not to have a CLI of some sort.

##### System requirements

* The system requirements for both the development and execution of this script are not very high, and it can be realistic to assume that any machine with the required OS and tools can do the job. However, it is worth noting that when harvesting large areas of the filesystem for metadata, that users may experience faster execution on machines with higher specs.

### Component 2: File Metadata Searcher

The metadata searcher will make use of a GUI and will enable the user to view the metadata for each file in the CSV file, which will be organised in a spreadsheet-like grid. There will be an ability to sort and filter the data by attribute values, and a search function which can take multiple criteria. The spreadsheet will have a degree of interactivity, in that double clicking on a record for a file will open it.

#### Metadata Searcher Technical Details

##### Language

The Metadata Searcher will be implemented using python, and for the GUI we will be making use of packages such as TKinter, PyQt ,Pyside etc. Our final choice on which one to use will come down to whichever package has all the required functionality we need.

##### Tools

* Computer running a Linux-based OS
* A text editor such as Atom, Vim, Sublime Text etc
* Latest version of Python will be used, for which an interpreter usually comes preinstalled on most popular Linux distributions

Most members of the team will most likely be using an IDE of some sorts, like PyCharm, Spyder etc.

##### Storage

It can be expected that the metadata searcher GUI program will have a larger storage footprint than the metadata harvester. That being said, it is expected that the metadata searcher program should not take up much storage space at all on any machine it is used on.

##### Software components

* Linux-based OS will be used for development and testing of the harvesting script and all required tools for development. It is most importantly required to run the script, as it will be designed to work with the Linux filesystems specifically.
* Latest version of Python will be used, for which an interpreter usually comes preinstalled on most popular Linux distributions
* At the minimum, we will be designing the searcher program to be invoked via the CLI. There is also an option of making it a standalone executable, which wont require much work, which will enable it to be executable like any other regular standalone program.

##### System requirements

* The system requirements for both the development and execution of this script are not very high, and it can be realistic to assume that any machine with the required OS and tools can do the job. However, it is worth noting that when harvesting large areas of the filesystem for metadata, that users may experience faster execution on machines with higher specs.

### Problems (Scope, Limitations, Assumptions)

During our initial meetings with our client, together we have come to define a scope for the project. Although this may be subject to change during the project, this will be our starting point to help outline the product we are making, and what we will and will not be implementing.

#### Metadata Harvester Scope

##### Scope Description:

In Scope:

* The script when executed will produce a CSV-formatted text as output detailing all files and their metadata within the user specified directory
* There will be an option available to specify a recursive directory search- which will enable the harvester script to harvest metadata from subdirectories of the target directory as well
* The script will only be accessible to invoke via the CLI
* The script will harvest the metadata of all file types found, and will be able to handle differing file types when formatting the CSV output
* Will be implemented to work with Linux OS distributions

Out of Scope:

* The script will not include options to produce output in any other formats
* The script will not be accessible via a GUI
* The script will not be designed to work with filesystems on other Operating Systems besides distributions of Linux

#### Metadata Harvester Potential Problems:

* There are many different file types, each with varying sets of attributes, some of which are not mandatory. This needs to be accounted for when formatting the output of the metadata harvester.
* Even files that are the same type, may have differing metadata attributes that have a value. An example of this is there may not be an author value for some text files, whereas there may be for others. This also needs to be taken into consideration.
* Due to time constraints on the project, a desired format for the output has to be agreed on as early as possible. Once a standard CSV format for output is decided than the team can also begin development on the Metadata Searcher, which will help avoid an unnecessary delay to development of the second component of this project.
* Should certain types of files be ignored? Examples of files that may be ignored are temporary files, this will need further clarification with our client.
* Should users only be able to harvest data for files of which they are the owners of? How will this be handled? What permissions will this script be executing under?

#### Metadata Harvester Limitations:

* Must be invoked via CLI
* Must only use open-source libraries
* Standard output must be CSV formatted to allow use of output by searcher program
* Must be implemented in Python
* Must be fully developed before or at the same time as the metadata searcher program to help facilitate the latter stages of testing.

#### Metadata Harvester Assumptions:

* Users should know how to use the CLI
* Users of the harvesting script should have authorised access to a Linux OS from which they can harvest file metadata
* Although the Harvester script doesn’t require a windowed environment, it’s main utility for the client comes from its pairing with the second component of this program, the metadata searcher, which requires a GUI and in turn a windowed environment.
* Users should not use the script for nefarious purposes and will only use them to harvest and search metadata for files that they have permission to access.

#### Metadata Searcher Scope

##### Scope Description:

In Scope:

* Searcher program will take as input the data produced from the Harvester Script only
* The Searcher program will be designed to allow user interaction via a GUI
* Files and their metadata will be displayed as records in a spreadsheet-like format
* Differing file types will be accounted for, resulting in sparsely populated fields
* File data displayed will be sortable by any metadata attribute (field)
* A search option will be implemented to enable searching my one or more attribute values, which results in only matching records being displayed
* Each file metadata record displayed will be clickable, which will then open the file with it’s default application for the file’s type (via a double click)
* Like the Harvester script, the Searcher must be usable on Linux.

Out of Scope:

* The Searcher Program will not be designed to take input from sources other than the Harvester Script
* The Searcher program will not be designed to offer interfaces other than the GUI
* There will not be an option to change the format of data displayed
* Files will only be able to be opened via double clicking their associated record, no other file operations will be possible. (Such as deletion, renaming etc.)
* The Searcher program will not be implemented to run on other OSes other than Linux distros.

#### Metadata Searcher Potential Problems:

* The harvested data used by the Searcher will have some inconsistencies, such as empty metadata fields, which will need to be accounted for
* How will the files and metadata displayed be organised by default?
* What will be the default order for file metadata attributes displayed?
* If files are sorted by a field how will others be sorted that do not have a value for that field?
* Will displayed files and metadata be organised by file type in separate sections? Or will they be displayed all together, sorted depending on another metadata attribute. (such as filename)
* The format of the CSV input that the searcher requires must have a format that is agreed upon as early as possible, as without this development of the searcher program will be delayed.
* Will the program be invoked via CLI or a standalone Launcher? Or both?

#### Metadata Searcher Limitations:

* Must make use of a GUI
* Must only use open-source libraries
* Must be implemented in Python

#### Metadata Searcher Assumptions:

* Users should know how to use the CLI
* Users should have access to a Linux system that is not headless in order to make use of the Searcher program and its GUI
* User will not use the programs for nefarious purposes and will only use them to harvest and search metadata for files that they have permission to access.
* Users will only attempt to use the Searcher with CSV data produced by the companion Metadata Harvester script developed by our team.

### Objectives

* Implement all functionality requested for by the client
* Make all deliverables on time and within budget
* Complete unit testing of all software components throughout development
* Complete program documentation before project closure

### Expected Outcome and deliverables

The expected outcome of this project is to provide both components with the full functionality desired by the client.

The specific deliverables will be:

1. The file metadata harvester program
2. The file metadata searcher program
3. Supporting documentation for both components

# 3 Project Design

Notes:

\_ Design specification depends on the type of projects

\_ For example, for a software development project design

{ Functional requirements

{ Non-functional requirements

{ High level (architecture) design

\_ For information system projects

{ Methodology design

{ Data collection strategy

{ Data analysis strategy

\_ The speci\_cation should depend on the agreement within the team and

supervisor

# 4 Work Break-down Structure and Task Scheduling

Notes:

\_ Deliverables and tasks should have reference back to the objectives speci\_ed

in \Project Specification"

{ Break the project plan into phases based on objectives, then

{ Break each phase into tasks, then

{ Break each task into sub-tasks if necessary

{ No need to go to further details

\_ Present the schedule of tasks and deliverables in the table or \_gure form

\_ Present the break-down structure in the table form.

\_ Ideally, the WBS and related chart(s), etc. should be generated using

project management tools, e.g., Microsoft Project.

# 5 Time and Cost Estimation

Notes:

\_ Cost of resources (e.g., hardware, software) required for project implemen-

tation

\_ Time estimation is based on the project plan and should have reference

back to the information speci\_ed in \Work Break-down Structure and Task

Scheduling"

\_ Labor Cost

{ Labor cost is based on the number of team members, roles, and the

time required to complete the project

{ Make up the hourly rates based on the current market standard

# 6 Project Schedule

Notes:

\_ GANTT Chart of the project schedule

\_ Speci\_ed based on the agreement within the team and the supervisor

\_ Specifying all milestones (the project lifecycle should be within this semester)

\_ Loss of data

# 7 Risk Management Plan

Notes:

\_ Specify and describe all potential risks

risks to the completion of project, e.g.,

\_ Project delay

\_ Loss of team member

\_ Loss of equipment

\_ Loss of data

\_ Loss of necessary services

\_ etc.

risks to professionalism and professional ethics, e.g.,

\_ leak of customers' privacy

\_ leak of client's business confidentiality

\_ leak of intellectual property like system design

\_ etc.

\_ Specify the risk with Probability (in percentage), Impact to the project

(in percentage), and Status (in High, Neutral, or Low Risk)

\_ Describe the risk management plan to deal with EACH of the specified

Risks

**Table 1. Risks to the Completion of the Project**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Risk and Consequences** | **Probability (%)** | **Impact (%)** | **Status** | **Mitigation Response** |
| 1.1 | Project delay | 70 | 90 | High | If project begins to fall behind, tempo of team zoom meetings to increase from weekly to bi-weekly, and if necessary, tri-weekly. |
| 1.2 | Loss of team member | 25 | 95 | Neutral | If team suffers loss of team member, client will be re-engaged to have their expectations massaged in accordance with new reality. |
| 1.3 | Loss of equipment | 40 | 20 | Low | Its not outside the realms of possibility that one of our team will have their laptop or phone die, but that’s no worries, they can easily replace. |
| 1.4 | Unable to Implement a working solution | 30 | 100 | High | If either our front-end or back-end teams are unable to implement a working solution, human resources from the other team will be reassigned to the team which is struggling. If both teams are struggling, CSC3400 course staff assets will be engaged to give us a hint. |

**Table 2. Risks to Professionalism and Professional Ethics**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Risk and Consequences** | **Probability (%)** | **Impact (%)** | **Status** | **Mitigation Response** |
| 2.1 | Loss of data | 60 | 50 | Neutral | If we lose data due to inexperience with github or for another reason, we just have to do it again, and take proper backups. |
| 2.2 | Loss of necessary services | 10 | 70 | Neutral | Slack and github are enterprise grade systems which are unlikely to go down. USQ study desk however is a different story... |
| 2.3 | Leak of customer’s privacy | 10 | 80 | Neutral | Loose lips sink ships. If any team member inadvertently leaks private customer details, they will be reminded of their importance of proper professional conduct. |
| 2.4 | Leak of client’s business confidentiality | 10 | 75 | Neutral | If any team member inadvertently leaks private customer details, they will be reminded of their importance of proper professional conduct. |
| 2.5 | Leak of intellectual property like system design | 10 | 90 | Neutral | If any team member inadvertently leaks private customer details, they will be reminded of their importance of proper professional conduct. |

# 8 Code of Conduct

Notes:

\_ Team principles

{ The principles of behaviour, communication, operational processes,

and professional ethics that the team agrees to abide for.

{ Principles should be high-level statements that describe what your

team considers to be the key values, beliefs and norms that contribute

to an e\_ective team environment;

{ Justify each of the principles for why all team members need to follow

it;

\_ Team communication and operational process

{ Operational processes should show how the principles are to be applied

to the daily operations during the project.

{ Provide at least 4-6 examples of communications and operational pro-

cesses that your team has agreed upon.

\_ Professionalism and professional ethics

{ the way that the team deals with customer privacy, business con\_-

dentiality and intellectual properties

{ the way (e.g., attitude) that the team deals with stakeholder, such as

client(s), customers, advisor(s), etc.

{ the way that the team members treat each other;

{ the way that the team deals with data;

{ use of development tools and resources, e.g., licensed vs. pirate soft-

ware, unauthorised resources such as data, images and sample prod-

ucts, etc.

{ provide at least 4-6 examples of professionalism and professional ethics

that your team has agreed upon.

\_ Non-compliance

{ Record your team's agreed de\_nitions of minor non-compliance (in

a way that may adversely a\_ect the project). You should provide

samples to help clarify the de\_nitions.

{ Record your team's agreed de\_nitions of major non-compliance (in a

way that has a major negative impact upon the team's success). You

should provide samples to help clarify the de\_nitions.

{ Separate the de\_nitions to initial and basic rules; minor and major

transgressions

Team name Page 8

Project Plan Version: version number

\_ Dispute resolution and conict management

{ State how your team has agreed to resolve issues like minor breaches

of this Agreement.

{ State how your team has to resolve issues like major breaches of this

Agreement.

# 9 Conclusions

Notes:

\_ Summarize the information presented in the document

\_ Make the conclusions to the document

# References

Sommerville, I. and Stevens, P. (2012). *Introduction to software engineering*. Frenchs Forest, N.S.W.: Pearson.

Rosenblatt, H. and Shelly, G. (2014). *Systems analysis and design, tenth edition*

# Appendix

# Contribution Statement

The undersigned members of this team agree to abide by this project plan to

ensure the successful completion of the project. The members also agree that the

contribution percentages specified below reflect the true level of contributions

made by each of the members to the works reported in this document.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Student Name | Student ID | Signature | Contribution % | Date |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |