**A Forensic Overview of Browser Cache Artefacts**

**Submitted in partial fulfilment of the requirements of**

**Edinburgh Napier University for the Degree of**

**BSc Cyber Security**

**School of Computing**

**April 2023**

# Authorship Declaration

I, Seth H, confirm that this dissertation and the work presented in it are my own achievement.

Where I have consulted the published work of others this is always clearly attributed;

Where I have quoted from the work of others the source is always given. With the exception of such quotations this dissertation is entirely my own work;

I have acknowledged all main sources of help;

If my research follows on from previous work or is part of a larger collaborative research project I have made clear exactly what was done by others and what I have contributed myself;

I have read and understand the penalties associated with Academic Misconduct.

I also confirm that I have obtained **informed consent** from all people I have involved in the work in this dissertation following the School's ethical guidelines

Signed:

Date: 12/04/2023

Matriculation no:

# General Data Protection Regulation Declaration

Under the General Data Protection Regulation (GDPR) (EU) 2016/679, the University cannot disclose your grade to an unauthorised person. However, other students benefit from studying dissertations that have their grades attached.

**The University may not make this dissertation available to others.**

Signed:

Date:

## Abstract

With technology rapidly evolving, Statista (2022) reports that 63.1% of the global population are accessing the internet using web browsers as of July 2022. These web browsers store a copy of the visited webpage’s resources (e.g., images and videos) using browser cache files. During a forensic investigation, these browser cache files can provide evidence to the types of content that a user has been viewing, and a variety of tools can be used to parse these browser cache files.

This project develops an MVP Cross-Browser Cache Parser through researching the knowledge and processes required for its development. A critical evaluation of the performance and capabilities of this MVP Cache Parser, ChromeCacheView, and MZCacheView is then performed to understand each parsers advantages and limitations.

The literature review critically evaluated contemporary literature and introduced knowledge on the cache mechanisms of Google Chrome and Mozilla Firefox. From this, the use of portable and private web browsing capabilities was identified as one of the key factors which affected the availability of cache files. Additionally, the literature review also revealed that the existing research on the cache mechanisms for Google Chrome and Mozilla Firefox was outdated. A comparison was then performed, and the latest cache structures for Chrome and Firefox were extracted. This research surrounding Chrome and Firefox is then taken forward and used to inform the experiment methods of the methodology.

Furthermore, the methodology outlines the methods which are supported by this project. The experiment methods define the MVP Cross-Browser Cache Parsers requirements using the MoSCoW prioritisation framework and the processes which will be followed during the cache parser’s development in Python. Using the processes and performance metrics outlined in the evaluation methods, the MVP Cross-Browser Cache Parser, ChromeCacheView, and MZCacheView are then evaluated.

The results of this evaluation revealed that the MVP Cross-Browser Cache Parser was able to parse the different cache mechanisms used by Google Chrome and Mozilla Firefox. Additionally, the outcome of this evaluation recommends using the MVP Cross-Browser Cache Parser over

ChromeCacheView when parsing Chrome cache files. For Firefox’s cache files, this project found that the MVP Cross-Browser Cache parser and MZCacheView should be used together.

## Contents

List of Tables ............................................................................................................................. 8

List of Figures ...........................................................................................................................10

List of Abbreviations ..................................................................................................................12 Glossary of Terms .....................................................................................................................13

1. Introduction ...........................................................................................................................17
   1. Background .....................................................................................................................17
   2. Aim ..................................................................................................................................19
   3. Research Questions ........................................................................................................19
   4. Objectives ........................................................................................................................19
   5. Rationale .........................................................................................................................19
2. Literature Review ..................................................................................................................20
   1. Introduction ......................................................................................................................20
   2. Forensic Science, Digital Forensics, and Browser Forensics ...........................................20
   3. Hypertext Transfer Protocol (HTTP) and HTTP Caching .................................................21
      1. The Hypertext Transfer Protocol ...............................................................................21
      2. HTTP Caching Mechanisms ......................................................................................22
   4. Portable and Private Web Browsing Capabilities .............................................................24
      1. Overview of Private Web Browsing Capabilities ........................................................24
      2. Overview of Portable Web Browsers .........................................................................25
      3. Past Investigations into Private and Portable Web Browser Capabilities ...................26
      4. Google Chrome – Incognito Mode and Chrome Portable ..........................................26
      5. Mozilla Firefox – Private Browsing and Firefox Portable ............................................29
   5. Web Browser Cache Mechanisms and Structures ...........................................................30
      1. Comparison of the Caching Mechanisms for Google Chrome and Mozilla Firefox.....30
      2. Comparison of the Cache Structure’s for Google Chrome and Mozilla Firefox ..........31
      3. Cache Parsers for Google Chrome and Mozilla Firefox .............................................37
   6. Summary .........................................................................................................................38
3. Methodology .........................................................................................................................39
   1. Introduction ......................................................................................................................39
   2. Quality Issues ..................................................................................................................39
      1. Reliability ..................................................................................................................39
      2. Validity ......................................................................................................................40
      3. Error ..........................................................................................................................40
      4. Bias ...........................................................................................................................40
      5. Ethical Considerations ..............................................................................................40
   3. Literature Review Methods ..............................................................................................41
   4. Project Management Methods .........................................................................................42
   5. Experiment Method .........................................................................................................42
      1. Introduction ...............................................................................................................42
      2. Cache Parser MoSCoW Requirements .....................................................................42
      3. Cache Parser Development ......................................................................................43
      4. Data Generation ........................................................................................................46
   6. Evaluation Methods .........................................................................................................47
      1. Selected Cache Parsers ............................................................................................48
      2. Performance Metrics Collected .................................................................................48
      3. Evaluation Process ...................................................................................................49
   7. Summary .........................................................................................................................49
4. Results and Discussion .........................................................................................................50
   1. Introduction ......................................................................................................................50
   2. Changes ..........................................................................................................................50
   3. Technical Artefact – MVP Cross-Browser Cache Parser .................................................51
   4. Quantitative Evaluation ....................................................................................................52
      1. Quantitative Evaluation Results .................................................................................52
      2. Quantitative Evaluation Discussion ...........................................................................54
   5. Qualitative Evaluation ......................................................................................................56
      1. Qualitative Evaluation Results ...................................................................................56
      2. Qualitative Evaluation Discussion .............................................................................60
   6. Results Conclusion ..........................................................................................................61
5. Conclusion ............................................................................................................................62
   1. Introduction ......................................................................................................................62
   2. Overview of The Project’s Conclusions ............................................................................62
   3. Project Reflection ............................................................................................................63
      1. Aim............................................................................................................................63
      2. Objectives .................................................................................................................63
      3. Research Questions (RQs) .......................................................................................65
   4. Limitations .......................................................................................................................65
   5. Future Work .....................................................................................................................66
   6. Self-appraisal ..................................................................................................................66
6. References ...........................................................................................................................68
7. Appendices ...........................................................................................................................73
   1. Appendix A – Chrome Cache Address Structure .............................................................73
   2. Appendix B – Reliability ...................................................................................................74
   3. Appendix C – Error ..........................................................................................................75
   4. Appendix D – Bias ...........................................................................................................75
   5. Appendix E – Reputable Source Repositories .................................................................76
   6. Appendix F – Keywords ...................................................................................................77
   7. Appendix G – Project Gantt Chart....................................................................................78
   8. Appendix H – MoSCoW Prioritisation Framework ............................................................79
   9. Appendix I – Cache Parser Libraries ...............................................................................80
   10. Appendix J – Multi-Browser Cache Identification Process..............................................81
   11. Appendix K – Chrome Cache Parsing Process ..............................................................83
   12. Appendix L – Firefox Cache Parsing Process ................................................................84
   13. Appendix M – Cache Recovery Mechanism Process .....................................................85
   14. Appendix N – CLI Integration Process ...........................................................................86
   15. Appendix O – Reporting Mechanism Process ................................................................87
   16. Appendix P – High-Level Object-Orientated Cache Parser Code Structure ...................88
   17. Appendix Q – Environment Specifications and Set-up Process .....................................89
   18. Appendix R – Standard Browser Installation Process ....................................................91
   19. Appendix S – Webpages Selected for Data Generation .................................................93
   20. Appendix T – Cache Parser Commands ........................................................................94
   21. Appendix U – Quantitative Evaluation Metrics ...............................................................95
   22. Appendix V – Qualitative Evaluation Questions .............................................................96
   23. Appendix W – Full Evaluation Process ..........................................................................97
   24. Appendix X – Windows Performance Monitor ................................................................98
   25. Appendix Y – Evaluation Process Changes ...................................................................99
   26. Appendix Z – The Cache Parser’s Command-Line Integration .................................... 100
   27. Appendix AA – The Cache Parser’s Cache Identification Functionality ........................ 102
   28. Appendix AB – The Cache Parser’s Chrome Parsing Mechanism ............................... 103
   29. Appendix AC – The Cache Parser’s Firefox Parsing Mechanism ................................. 106
   30. Appendix AD – Examples of The Cache Parser’s Cache Structures ............................ 109
   31. Appendix AE – The Cache Parser’s Recovery Mechanism .......................................... 110
   32. Appendix AF – The Cache Parser’s Reporting Mechanism.......................................... 112
   33. Appendix AG – The Cache Parser’s Error Handling and Logging ................................ 113
   34. Appendix AH – Full Quantitative Evaluation Results .................................................... 115

## List of Tables

Table 1 – List of Abbreviations ..................................................................................................12

Table 2 – Glossary of Terms .....................................................................................................13

Table 3 – Table comparing the similarities and differences between the cache folder contents for

Google Chrome and Mozilla Firefox - (Habben, 2015; Suma, Dija, & Pillai, 2017) .....................32

Table 4 – Comparison of Chrome’s index file structure - red names indicate fields that weren’t present in the research conducted by Suma et al. (2017), but were included in the official source

code provided by Chromium (2022a) ........................................................................................33

Table 5 - Comparison of Chrome’s cache entry structure - red names indicate fields that weren’t present in the research conducted by Suma et al. (2017), but were included in the official source

code provided by Chromium (2022a) ........................................................................................34 Table 6 – Comparison of Firefox’s index file’s header structure - red names indicate fields that weren’t present in the research conducted by Habben (2015), but were included in the official

source code provided by Searchfox (2022) ...............................................................................35

Table 7 – Comparison of Firefox’s index file’s record structure - red names indicate fields that weren’t present in the research conducted by Habben (2015), but were included in the official

source code provided by Searchfox (2022) ...............................................................................36 Table 8 – Comparison of Firefox’s entry file metadata header structure - red names indicate fields that weren’t present in the research conducted by Habben (2015), but were included in the official

source code provided by Searchfox (2021) ...............................................................................36 Table 9 – Cache Parser MoSCoW Requirements .....................................................................43

Table 10 – Publicly available cache parsers selected for the evaluation. ..................................48 Table 11 – Quantitative evaluation results showing the total number of cached webpage resources identified and recovered by the MVP cross-browser cache parser, ChromeCacheView,

and MZCacheView. ...................................................................................................................52

Table 12 – Quantitative evaluation results showing the performance metrics collected after using the MVP Cross-Browser Cache Parser and ChromeCacheView to parse chrome cache files. ..53 Table 13 – Quantitative evaluation results showing the performance metrics collected after using the MVP Cross-Browser Cache Parser and MZCacheView to parse firefox cache files. ...........54 Table 14 – List of browsers which are supported by each of the cache parsers. .......................56

Table 15 – Comparison of the report file formats supported by the MVP Cross-Browser Cache

Parser, MZCacheView, and ChromeCacheView. ......................................................................57 Table 16 – Comparing the information present in Chrome’s cache files, to the information extracted and outputted by ChromeCacheView and the MVP Cache Parser. ...........................58

Table 17 – Comparing the information present in Firefox’s cache files, to the information extracted and outputted by MZCacheView and the MVP Cache Parser. ..................................................59

Table 18 – Structure of Chrome’s cache addresses - (Chromium, 2022b) ................................73

Table 19 – Actions that are to be taken to consider reliability in each of the methods. ..............74

Table 20 – Actions that are to be taken to minimise the risk of error. ........................................75

Table 21 – Actions that will be taken to account for bias in the self-generated test data. ...........75

Table 22 – Reputable source repositories used during the literature review. .............................76

Table 23 – Keywords used to search and filter for sources during the literature review. ............77

Table 24 – Categories of the MoSCoW Prioritisation Framework ..............................................79

Table 25 - List of python libraries that will be used for the cross-browser cache parser. ............80

Table 26 – Regular Expressions that will be used to identify cache folders for Chrome and Firefox

.................................................................................................................................................82

Table 27 – Virtual Environment Specifications ..........................................................................90

Table 28 – Versions of Google Chrome and Mozilla Firefox used during data generation. ........92 Table 29 – Table of websites visited to generate the data used for the evaluation of the cache

parsers. .....................................................................................................................................93

Table 30 – Example commands that will be used to execute the individual cache parsers. .......94

Table 31 – Cache folder paths that will be used during the evaluation. .....................................94 Table 32 – Performance-based metrics collected from the cache parsers during the evaluation.

.................................................................................................................................................95

Table 33 – Comparative questions asked to qualitatively evaluate the capabilities of the cache

parsers. .....................................................................................................................................96

Table 34 – The actual commands that were used to execute the cache parsers during the

evaluation. ................................................................................................................................99

Table 35 – The actual cache folder paths that were used during the evaluation. .......................99 Table 36 – Full results for the quantitative evaluation of the MVP Cross-Browser Cache Parser and ChromeCacheView against chrome cache files. .............................................................. 115 Table 37 – Full results for the quantitative evaluation of the MVP Cross-Browser Cache Parser

and MZCacheView against firefox cache files. ........................................................................ 116

## List of Figures

Figure 1 – Desktop Web Browser Statistics by (StatCounter, 2022) ..........................................17

Figure 2 – Client HTTP GET Request - (Fielding, Nottingham, & Reschke, 2022a) ..................21

Figure 3 – Server HTTP Response - (Fielding, Nottingham, & Reschke, 2022a) .......................22

Figure 4 – Cache Response and Cache Request Directives - (Fielding, Gettys, Mogul, H, &

Berners-Lee, 1997) ...................................................................................................................22

Figure 5 – Example of the first request sent being cached by the client’s web browser. ............23

Figure 6 – Example of the same request being performed; this time being fetched from the

browser’s local cache. ...............................................................................................................23

Figure 7 – Google Chrome normal browsing mode – (Nelson, Shukla, & Smith, 2019) .............27

Figure 8 – Google Chrome Incognito Browsing mode – (Nelson, Shukla, & Smith, 2019) .........27

Figure 9 – Extracted Chrome Data – “X” indicates this information was able to be extracted, “-“ means this information was not able to be extracted (Nelson, Shukla, & Smith, 2019). .............27 Figure 10 – Artefacts extracted from different browsers running in private mode – (Flowers,

Mansour, & Al-Khateeb, 2016) ..................................................................................................28

Figure 11 – Artefacts extracted from different portable browsers running in normal mode –

(Flowers, Mansour, & Al-Khateeb, 2016)...................................................................................28

Figure 12 – Firefox artefact recovery and location – (Nelson, Shukla, & Smith, 2019)...............29

[Figure 13 - Google Chrome's cache folder (left) provided by (Suma, Dija, & Pillai, 2017) and](https://livenapierac-my.sharepoint.com/personal/40510741_live_napier_ac_uk/Documents/Module%2017%20-%20Synoptic%20Project/Chapter%206%20-%20Final%20Drafts/Final_Draft_Feedback.docx#_Toc132184147)

[Mozilla Firefox's cache folder (right) provided by (Habben, 2015). ............................................31](https://livenapierac-my.sharepoint.com/personal/40510741_live_napier_ac_uk/Documents/Module%2017%20-%20Synoptic%20Project/Chapter%206%20-%20Final%20Drafts/Final_Draft_Feedback.docx#_Toc132184147)

Figure 14 – Literature review source selection process. ............................................................41 Figure 15 – Flowchart defining the process taken to generate the cache files used during the

evaluation of the cache parser tools. .........................................................................................47 Figure 16 – Project Gantt Chart – Methodology Progress .........................................................78

Figure 17 – Cache Identification Process. .................................................................................81

Figure 18 – The logic and processing flow behind the Chrome Cache Parser ...........................83

Figure 19 – The logic and processing flow behind the Firefox Cache Parser ............................84 Figure 20 – The logic and processing flow behind decompressing the raw bytes of the requested

resource. ...................................................................................................................................85 Figure 21 – Cache Parser CLI Integration Process ...................................................................86

Figure 22 – Reporting Mechanism Process ...............................................................................87 Figure 23 - High-level overview of the cache parser's code structure, and the OO classes used.

.................................................................................................................................................88 Figure 24 – The environment set-up process used to create all the project’s virtual environments.

.................................................................................................................................................89

Figure 25 – The full process used to install Google Chrome and Mozilla Firefox onto the data

generation environment. ...........................................................................................................91

Figure 26 - The evaluation process used to evaluate the performance of the cache parsers. ....97 Figure 27 – The process used to collect performance metrics for each of the cache parsers during

the evaluation............................................................................................................................98

Figure 28 – The cache parser code used to implement the CLI Integration process. .............. 100

Figure 29 – The cache parser’s help message displaying the arguments and flags it accepts.101 Figure 30 – The cache parser code used to implement the multi-browser cache identification

process. .................................................................................................................................. 102

Figure 31 – The code used to control the parsing of Chrome’s index files and cache entries. . 103

Figure 32 - The code used to parse Chrome's index file. ......................................................... 104

Figure 33 - The code used to parse Chrome's cache entries. ................................................. 105

Figure 34 – The code used to control the parsing of Firefox’s index files and cache entry files.

............................................................................................................................................... 106

Figure 35 – The code used to parse Firefox’s index file. ......................................................... 107

Figure 36 – The code used to parse Firefox’s cache entries. .................................................. 108

Figure 37 - Example of the cache structures defined in the cache parser's code..................... 109

Figure 38 – The code used to recover and recreate the cached webpage resources for Google

Chrome. .................................................................................................................................. 110

Figure 39 – The code used to organise and output the recovered cached webpage resources.

............................................................................................................................................... 111

Figure 40 – The code used to format the extracted cache information into a report. ............... 112

Figure 41 – The code used to initialise the cache parser’s logging and output status updates to

the log file. .............................................................................................................................. 113 Figure 42 - An example of the cache parser's log file. ............................................................. 114

## List of Abbreviations

*Table 1 – List of Abbreviations*

|  |  |
| --- | --- |
| **Abbreviations** |  |
| ACM | Association for Computing Machinery |
| CLI | Command Line Interface |
| CSV | Comma-Separated Values |
| DSDM | Dynamic System Development Method |
| GB | Gigabyte |
| GUI | Graphical User Interface |
| HTTP | Hypertext Transfer Protocol |
| IEEE | Institute of Electrical and Electronics Engineers |
| IETF | Internet Engineering Task Force |
| ISP | Internet Service Provider |
| JSON | JavaScript Object Notation |
| LRU | Least Recently Used |
| MVP | Minimum Viable Product |
| NAT | Network Address Translation |
| OO | Object Orientation |
| RFC | Request For Comments |
| RQ | Research Question |
| URI | Uniform Resource Identifier |
| URL | Uniform Resource Location |
| USB | Universal Serial Bus |
| VM | Virtual Machine |

## Glossary of Terms

*Table 2 – Glossary of Terms*

|  |  |
| --- | --- |
| **Term** | |
| Browser Cache  (Also referred to as  HTTP Cache) | These are files which are stored by a browser on a user’s local system through a defined cache mechanism, and when parsed, can be used to recover videos, images, and scripts from webpages that the user has visited (Suma, Dija, & Pillai, 2017; Sawicki, Zych, & Sawicki, 2021). |
| Browser Forensics | The process of collecting forensic artefacts left by a user’s web browsing activity (Suma, Dija, & Pillai, 2017; Jadhav & Meshram,  2018). |
| Browsing Activity | A series of events (or activities) performed by a user, such as visiting a certain website, watching specific videos, and downloading files which are saved by the browser. |
| Cache Address | Cache addresses are found in chrome’s cache files, and they point to specific cache entries. |
| Cache Entries | Cache Entries are found in chrome’s cache files and contain information relating to the items cached by the browser. |
| Cache Mechanisms | The process of storing HTTP responses which contain the requested resources on the user’s local system, so that it can be easily re-used in later requests (Sawicki, Zych, & Sawicki, 2021). |
| Cache Parser | A small utility that finds and reads the cache folder of a web browsers, and parses each of the cache files to extract and display information relating to the files currently stored locally in the cache (NirSoft, 2022a; NirSoft, 2022b). |
| Cache Structures | The byte structure of an individual browser cache file. |
| Cache2 | Mozilla Firefox’s specific cache mechanism. |

|  |  |
| --- | --- |
| Criminal Investigation | This investigation involves the analysis of evidence taken from a crime to inform the outcome of a case, and subsequently, have the criminals apprehended. |
| Cyber Crimes | Any type of criminal activity that is either enabled-by or involved a computer or a network (Brush, Rosencrance, & Cobb, 2021). |
| Data File | A cache file found in Google Chrome’s cache folder. |
| Digital Forensics | A branch of forensic science that involves the identification, collection, and examination of digital data used in criminal investigations to help investigate cyber-crimes and identify evidence relating to these computer-assisted crimes (Sadiku, Tembely, & Musa, 2017; Hughes, et al., 2021). |
| Disk Cache | Chromium specific cache mechanism which is used by Google Chrome. |
| Entry File | A cache file found in Mozilla Firefox’s cache folder. |
| External File | A cache file found in Google Chrome’s cache folder. |
| Forensic Artefacts | The files of interest that can function as pieces of evidence for the investigator and that allow for the identification of user’s activities. |
| Forensic Investigator | An individual who analyses the extracted digital evidence when performing digital forensics (Studiawan, Sohel, & Payne, 2019). |
| Forensic Science | An investigatory tool used in criminal investigations which enables the analysis of evidence through the application of scientific knowledge and methodologies (Afridi, 2021). |
| HTTP Request | A message sent by a client to request a resource from a server. |
| HTTP Response | A message sent by a server in response to a HTTP request, either delivering the requested resource or providing information on why the request failed. |
| Hypertext Transfer  Protocol (HTTP) | The underlying protocol used by the Worldwide World to exchange messages across an established connection using a request/response module. Where clients construct request |
|  | messages to communicate with a server, and the server responds to these requests using response messages (Mozilla, 2022; Fielding, Nottingham, & Reschke, 2022a). |
| Index File | A cache file found in both Google Chrome’s and Mozilla Firefox’s cache folders. |
| Metadata | Additional information which describes other data. |
| Portable Web Browser | These are browsing applications which are stored and ran from an external storage device (Hariharan, Thakar, & Sharma, 2022; Flowers, Mansour, & Al-Khateeb, 2016). |
| Private Web Browsing  Capabilities | These are found embedded within common web browsers and allow a user to minimise the number of files stored by the browser on their device that contain information relating to their web browsing activities (Horsman, et al., 2019). |
| Webpage Resources | Files found on a webpage and that are used by a webpage (such as scripts, images, and videos) |
| Web Browser | An application which allows people to access websites via the internet to search for information and communicate with others (Flowers, Mansour, & Al-Khateeb, 2016). |

## Acknowledgements

Firstly, I’m extremely grateful to all of those who provided me support during this project in its entirety.

Specially thanks to John Colvin for the feedback and support given during the project.

Also, thanks should go to my workplace team who continuously supported me by providing me with the tools and expertise needed to perform the research conducted in this project.

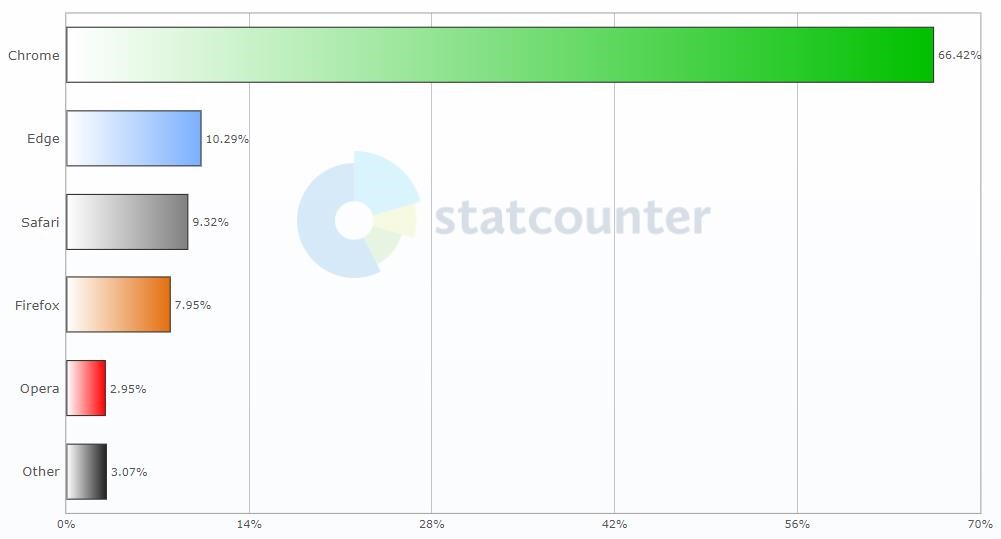
Lastly, I’d like to mention my family as they have supported me throughout this project by keeping my motivations high.

## 1. Introduction

### 1.1 Background

With technology evolving, a substantial percentage of the global population now have access to the internet, with Statista (2022) reporting that around 63.1% of the global population use the internet as of July 2022 (5.03 billion users).

According to StatCounter (2022), Google Chrome dominates the 2022 global desktop browser market, ahead of other commonly available web browsers (Figure 1).



*Figure 1 – Desktop Web Browser Statistics by (StatCounter, 2022)*

These web browsers store browser files on the user’s system which can be analysed to identify a user’s browsing activities (Flowers, Mansour, & Al-Khateeb, 2016). A forensic investigator will use browser forensics to acquire and analyse these browser files generated by the web browsers, which include, but are not limited to, browser history, cache, bookmarks, and cookies (Suma, Dija, & Pillai, 2017). These browser files are a vital part of digital forensics, and thus are also known as forensic artefacts.

Focusing more on the browser cache, it is an important forensic artefact because it enables forensic activities such as video reconstruction using the cached media content. Therefore, this allows a forensic investigator to discover and reconstruct the videos that a suspect had been viewing prior to the acquisition of the device (El-Tayeb, Taha, & Fayed, 2022).

However, the availability of forensic artefacts such as browser cache differs between web browsers and their embedded privacy features. To preserve their user’s privacy, Google Chrome has Incognito Mode, which is a form of private browsing integrated into the web browser that disables both caching mechanisms and browser history (Narayanan, Rajkumar, & Sobhana, 2017). Nevertheless, if a user has used private browsing modes, the user’s browsing history can still be retrieved from external entities such as Internet Service Providers (ISPs) or Employers (Google, 2022). Moreover, when private browsing modes are used, the extraction of cache files becomes infeasible as these embedded privacy features either disable the browser’s caching mechanism or deletes the cache once the private browsing session has been closed.

Portable web browsers store the web browser on an external storage device instead of the user’s computer (Flowers, Mansour, & Al-Khateeb, 2016). This can prove to be problematic because forensic artefacts of interest could potentially be stored on the external device, rather than the user’s local system. Without access to this external device, a forensic investigator has no way of extracting the user’s cache artefacts.

Consequently, both portable web browsers and embedded privacy features reduce the number of forensic artefacts left available on the user’s system. Therefore, the importance of the browser’s cache and a user’s desire to hide their browsing activities makes accurately parsing these forensic artefacts crucial. With the right tools, a user’s web browser cache can be parsed to recover cached webpage resources (e.g., images and videos) which relate to the user’s browsing activity.

Today, applications such as ChromeCacheView and MZCacheView facilitate the parsing of both Google Chrome and Mozilla Firefox caches, respectively. Both tools are publicly available and can be used to extract information relating to their respective browsers. However, these cache parsers are limited by nature as each tool only supports the parsing of one cache mechanism and only displays a subset of the information that can be extracted from the cache (Horsman, 2018b; Shafqat, 2016).

Therefore, this project addresses the limitations of publicly available cache parsers to allow forensic investigators to use one tool to recover a suspect’s cached files for multiple windowsbased web browsers which use different caching mechanisms.

### 1.2 Aim

The aim of this project is to develop a minimum viable product (MVP) cross-browser cache parser – focusing on the normal installations of two major windows-based web browsers: Google Chrome and Mozilla Firefox.

### 1.3 Research Questions

Supporting the aim, this report tries to answer the following research questions:

1. How do the cache structures and mechanisms present in Google Chrome and Mozilla

Firefox differ?

1. What information can be extracted from the caches of Google Chrome and Mozilla Firefox, using their normal, base installation?
2. In terms of capability and performance, how do publicly available cache parsers differ to the MVP cross-browser cache parser?

### 1.4 Objectives

To achieve the project’s aim, the following objectives will be completed:

1. Critically review contemporary literature surrounding the Hypertext Transfer Protocol (HTTP), HTTP Caching Mechanisms, Portable and Private Web Browser Capabilities, and the Cache Mechanism, Structures, and Parsers for Google Chrome and Mozilla Firefox.
2. Research and develop an experimental methodology based on the outputs of the literature review.
3. Execute the primary research and present the data needed to develop the MVP cache parser so that it can parse cache structures used in Google Chrome and Mozilla Firefox.
4. Develop an MVP cache parser to recover the web browser’s cached webpage resources.
5. Critically evaluate and discuss the performance and capabilities of the MVP cross-browser cache parser against publicly available cache parsers.
6. Conclude the findings of the project while reflecting and making recommendations based on the findings.

### 1.5 Rationale

Current cache parsers are limited in their ability to extract and analyse the different caching mechanisms present in multiple browsers. The ability to produce a single cache parser that fully supports multiple browsers and their different caching mechanisms would have positive impacts on both the academic sphere and forensic sector. In the forensic sector, a single tool that can parse different browser caches allows for a lightweight and efficient solution to extracting a user’s browsing activity. Whereas in the academic sphere, the development of this cache parser enables the opportunity to develop current academic research into the latest versions of Google Chrome and Mozilla Firefox, further outlining how cache is stored in these browsers and what information can be extracted. For this project, the normal installations of the most popular consumer-based web browsers have been selected as a pragmatic approach to resource restrictions.

## 2. Literature Review

### 2.1 Introduction

The objective of this literature review is to critically review contemporary literature surrounding HTTP Concepts, HTTP Caching Mechanisms, Portable and Private Web Browser Capabilities, and finally, the Cache Mechanisms, Structures, and Parsers present for Google Chrome and Mozilla Firefox. To achieve this, a thematic literature review will be conducted and will follow a defined structure.

Initially, an overview of the different branches of forensics will be provided, followed by a critical review of literature surrounding HTTP and the HTTP Caching Mechanisms. After this, existing literature surrounding Private and Portable Web Browsing Capabilities will be critically analysed to provide an insight into how these capabilities may affect the availability of browser cache. Finally, the thematic literature review critically analyses literature surrounding the caching mechanisms, structures, and publicly available parsers for both Google Chrome and Mozilla Firefox, providing the foundational knowledge needed for the experimental methodology.

In doing this, the literature review explores the problem, summarises relevant theory, and reviews existing work.

### 2.2 Forensic Science, Digital Forensics, and Browser Forensics

Defined by Afridi (2021), forensic science is an investigatory tool used in criminal investigations to enable the analysis of evidence through the application of scientific knowledge and methodologies.

Linking this to digital forensics, Hughes et al. (2021) and Sadiku, Tembely, & Musa (2017) describes digital forensics as a branch of forensic science that involves the identification, collection, and examination of digital data used in criminal investigations to help investigate cybercrimes and identify evidence relating to these crimes.

Jadhav & Meshram (2018) and Suma, Dija, & Pillai (2017) identify that browser forensics plays a crucial role in performing digital forensics during a criminal investigation. Both sources state that browser forensics is the process of collecting forensic evidence left by a user’s web browsing activity.

### 2.3 Hypertext Transfer Protocol (HTTP) and HTTP Caching

#### 2.3.1 The Hypertext Transfer Protocol

As covered by Mozilla (2022) in an article on the evolution of HTTP, Tim Berners-Lee developed the first version of HTTP between 1989 and 1991. This was originally used to exchange documents over the internet using a simple web browsing client.

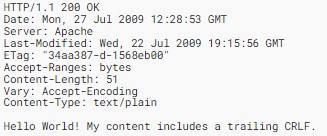
Defined by Berner-Lee, Fielding, & Frystyk (1996), HTTP/1.0 provided the foundational elements which included status codes to recognise success or failure, HTTP headers that allowed for the exchange of additional client/server information, and the first ever caching mechanism.

Shortly after this, HTTP/1.1 was introduced and since its introduction, HTTP/1.1 has been continuously refined and developed for over 20 years.

In the latest internet standard for HTTP, Fielding, Nottingham, & Reschke (2022a) define HTTP/1.1 as being a request/response protocol that is used for exchanging messages across an established connection. During this, the client constructs HTTP request messages with different headers to communicate and retrieve resources held by the server (Figure 2). Then, the server acts on these requests and responds by sending back a HTTP response message containing a status code, additional response metadata, and the requested resource’s content (Figure 3).



*Figure 2 – Client HTTP GET Request - (Fielding, Nottingham, & Reschke, 2022a)*



*Figure 3 – Server HTTP Response - (Fielding, Nottingham, & Reschke, 2022a)*

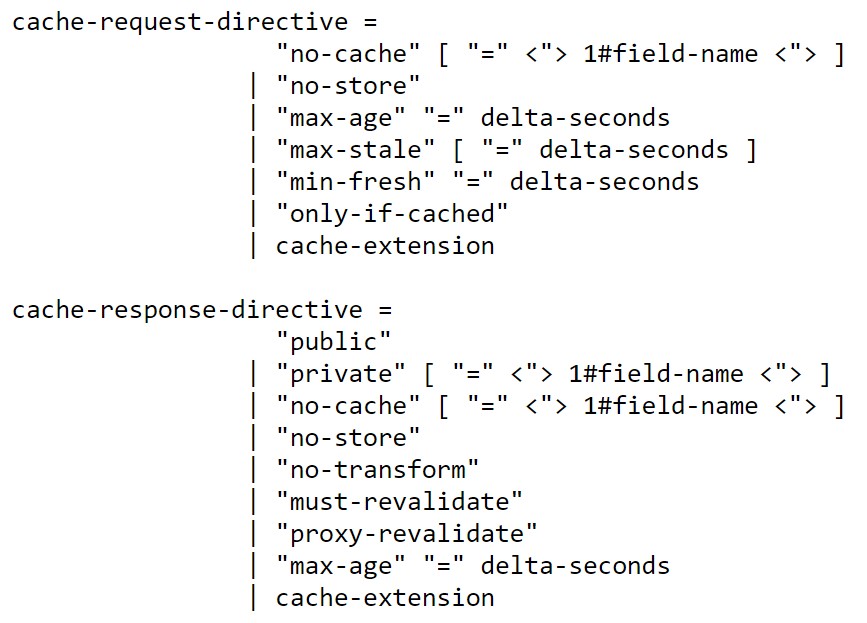
#### 2.3.2 HTTP Caching Mechanisms

As previously mentioned, HTTP/1.0 first introduced the concept of cache. Defined by Berner-Lee,

Fielding, & Frystyk (1996), cache is “A program’s local store of response messages and the subsystem that controls its message stored, retrieval, and deletion. A cache stores cachable responses to reduce the response time and network bandwidth consumption on future, equivalent requests.”

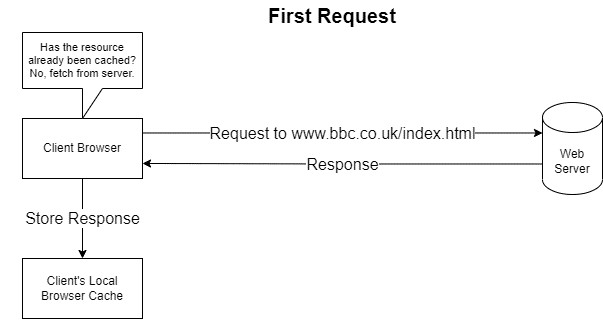
To understand what influences the HTTP caching mechanisms, Berner-Lee, Fielding, & Frystyk

(1996) explain that HTTP/1.0 originally used the “Pragma” header field to control the HTTP cache. However, this changed in HTTP/1.1, where additional cache control mechanisms were introduced to allow for better control. Fielding, Nottingham, & Reschke (2022b) defines these cache control mechanisms as a way for a server or client to explicitly define cache controls using the “cachecontrol” header and its subsequent directives. These directives override the default caching mechanism and effect the way that resources may be cached on a user’s system (Figure 4).



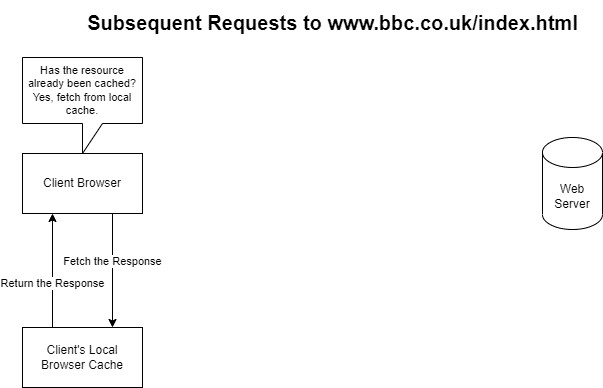
*Figure 4 – Cache Response and Cache Request Directives - (Fielding, Gettys, Mogul, H, & Berners-Lee, 1997)*

Alongside these internet standards, Sawicki, Zych, & Sawicki (2021) perform an analysis into the current security status of the local HTTP caching mechanisms. During this, they find that the improvement in efficiency caused by cache is achieved through storing the requested static resources of a webpage (e.g., images, and media) in a location that is more accessible for the client. Thus, the cache mechanisms store the responses containing the static resources locally so they can be re-used again in response to later requests (Figure 5 and Figure 6).



*Figure 5 – Example of the first request sent being cached by the client’s web browser.*

*Figure created by author using (diagrams.net, 2023).*



*Figure 6 – Example of the same request being performed; this time being fetched from the browser’s local cache.*

*Figure created by author using (diagrams.net, 2023).*

Supporting the research into HTTP cache conducted by Sawicki et al. (2021), Nguyen, Lo Iacono, & Federrath (2019) analyse the individual cache properties in more detail. In doing this, they identified that the “Cache-Control” header of a HTTP Response and Request can have a direct impact on how the cache is stored on a user’s local system. When coupled with the ‘no-store’ directive, this will prevent the cache from storing requested content.

The research performed by Nguyen et al. (2019) and Sawicki et al. (2021) is based on RFC 7234, which as of July 2022 obsolete and has been replaced by RFC 9111. However, both sources still accurately outline and emphasise the main concepts of HTTP caching which are relevant to this project.

Relating these caching concepts to browser forensics, Horsman (2018a) elaborates on the importance of HTTP cache and its ability to reveal cached resources of webpages that users have previously visited. Therefore, this makes understanding cache and what affects the cache critical, as it is a vital source of evidence in digital forensics.

### 2.4 Portable and Private Web Browsing Capabilities

Following on from understanding HTTP and HTTP caches, this section aims to highlight how the private and portable browsing capabilities of a web browser can drastically affect the digital forensics process by reducing the number of forensic artefacts available on a user’s system.

#### 2.4.1 Overview of Private Web Browsing Capabilities

As defined by Horsman et al. (2019), private browsing capabilities allow a user to minimise the amount of evidence stored on their devices that contain information relating to their web browsing activities.

Hasan et al. (2021) and Hughes et al. (2021) identified that the growth of private browsing modes began in the early 2000’s due to the growing concern of user privacy when browsing the internet. Safari was at the forefront of this when in 2005 it became the first web browser to offer extensive user privacy through disabling certain features that were tracking the user’s browsing activities.

Other web browsers sought competitive parity and began to implement their own private browsing modes. Today, privacy browsing modes are embedded within almost any publicly available web browser, each with their own implementation.

Using Google Chrome and Mozilla Firefox as an example, Chrome supplies “Incognito Mode” and Firefox supplies “Private Browsing” mode.

Google (2022) states: “When you browse privately, other people who use the device won’t see your history. Chrome doesn’t save your browsing history or information entered in forms. Cookies and site data are remembered while you’re browsing but deleted when you exit Incognito mode. You can choose to block third-party cookies when you open a new incognito window.”

Additionally, Mozilla (2022) states: “Private Browsing does not save your browsing information, such as history and cookies, and leaves no trace after you end the session.”

However, both browsers highlight the fact that a users’ browsing activity might still be visible, and that it does not make you completely anonymous on the Internet. Entities such as ISPs, Websites, or Employers may still be able to record a users’ browsing activity.

Supporting this, Hughes et al. (2021) investigated different browsers’ private modes to identify whether the browsers’ actually offer any additional privacy to their users. During their research, they investigate the private browsing modes of Google Chrome, Mozilla Firefox, Brave, and Microsoft Edge using both disk and memory forensics. Based on the results, they discussed that even with browser privacy features enabled, this does not always guarantee privacy. Their results also showed that Google Chrome offered the most privacy to its users through less browser artefacts being left behind when using Incognito Mode.

#### 2.4.2 Overview of Portable Web Browsers

As well as supporting previous definitions of private browsing, Flowers et al. (2016) and Hariharan, Thakar, & Sharma (2022) cover the concept of portable web browsers through analysing which artefacts are present when using portable web browsers.

Both identify portable browsers as being a browser application that is stored and ran from an external storage device (such as a USB stick). These portable browsers claim to provide added user privacy through storing its browser files on the external storage device that the browser was originally run from, leaving no trace on the user’s host system.

As mentioned by Hariharan et al. (2022), this can prove challenging for forensic investigators when they want to recover browser files to identify a user’s browsing activities. If the external device that the browser was executed from cannot be obtained, then theoretically, obtaining the files needed to extract cached webpage resources from a user’s browsing sessions is infeasible.

#### 2.4.3 Past Investigations into Private and Portable Web Browser Capabilities

When investigating the impact private browsing modes has on the availability of forensic artefacts,

Nelson, Shukla, & Smith (2019) investigate the forensic artefacts stored on a user’s system when using the private browsing modes of Google Chrome and Mozilla Firefox. Whereas Flowers et al. (2016) investigate whether the claims of privacy through portable web browsers are true by investigating artefacts left by portable browsers, and their private browsing modes.

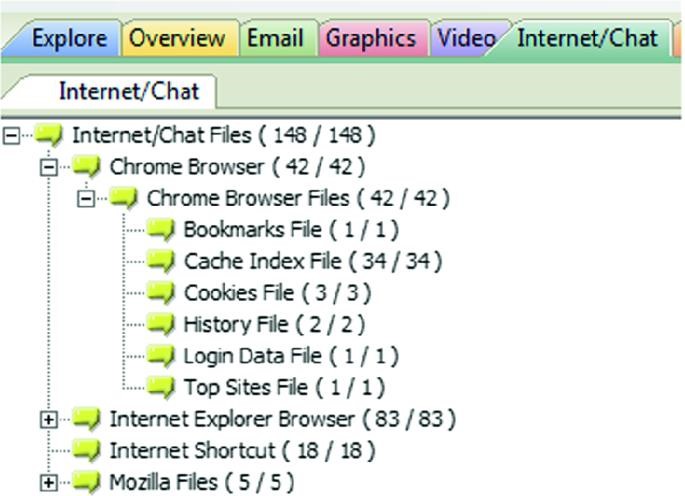
One downside to their research is that they both use outdated versions of each browser. Nelson, Shukla, & Smith (2019) use a version of Google Chrome which was released in September 2017, and a version of Mozilla Firefox released in August 2017. While Flowers et al. (2016) use a version of Google Chrome released in March 2015, and a version of Mozilla Firefox released in February 2015.

Horsman et al. (2019) highlighted this risk by stating that research into the private browsing capabilities of web browsers must be continuous as the vendors are constantly evolving their implementations to enhance user experience. Thus, suggesting that older versions of the software may differ to the newer versions.

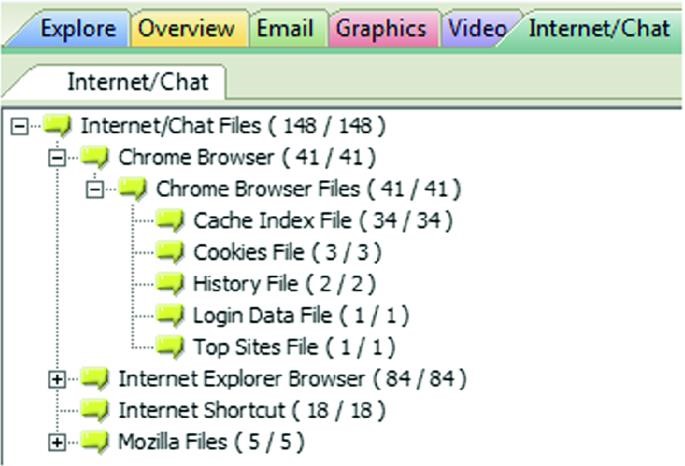
Even though their research is outdated, the results of their investigation can still be used reflect on the affect private and portable web browser capabilities might have on the availability of browser artefacts.

#### 2.4.4 Google Chrome – Incognito Mode and Chrome Portable

During their research, Nelson et al. (2019) found that there were some differences in the number of files stored, and more specifically, the file content that was stored between the normal and private browsing modes of Google Chrome (Figure 7 and Figure 8).

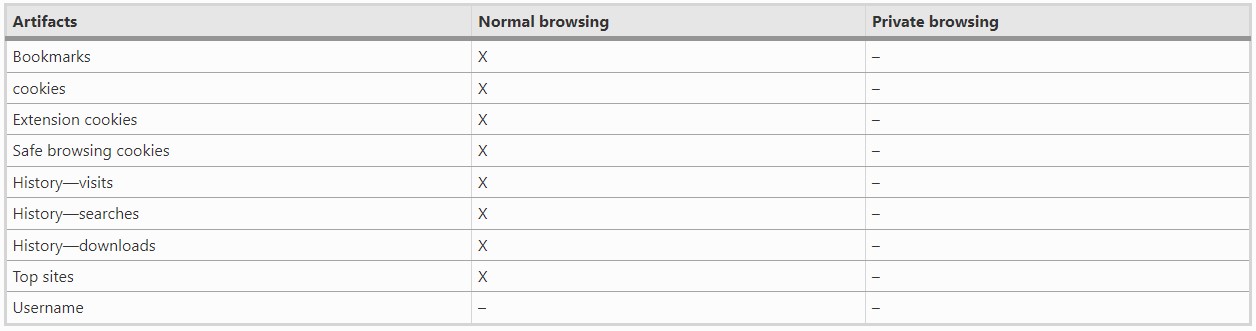


*Figure 7 – Google Chrome normal browsing mode – (Nelson, Shukla, & Smith, 2019)*



*Figure 8 – Google Chrome Incognito Browsing mode – (Nelson, Shukla, & Smith, 2019)*

When investigating the content of these recovered artefacts, Nelson et al. (2019) found that some of the recovered file’s contents was missing. Therefore, they produced a table to summarise the difference in extracted data between normal and private browsing modes (Figure 9).

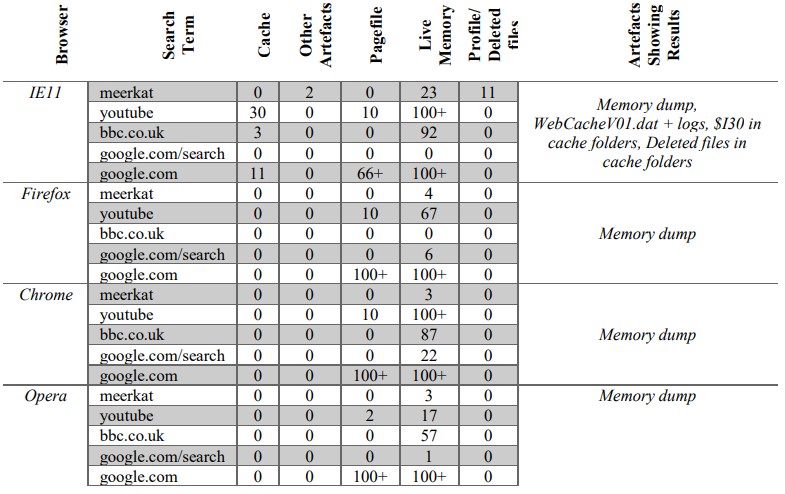


*Figure 9 – Extracted Chrome Data – “X” indicates this information was able to be extracted, “-“ means this information was not able to be extracted (Nelson, Shukla, & Smith, 2019).*

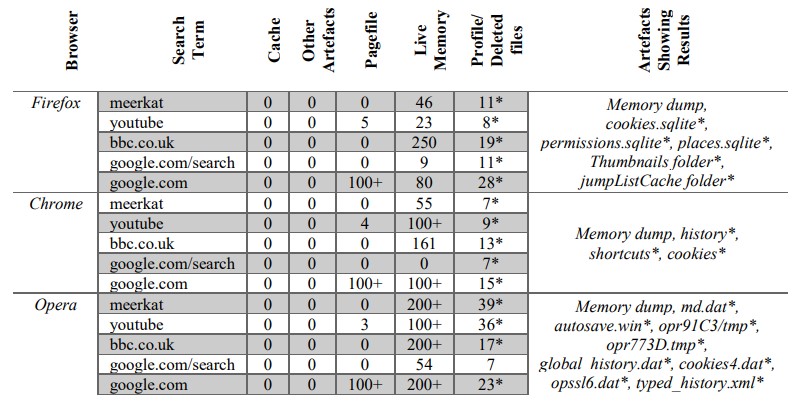
In support of the investigation conducted by Nelson et al. (2019), Flowers et al. (2016) investigated the forensic artefacts available after using the private and portable capabilities of Google Chrome, with parts of their investigation focusing on chrome cache files.

Flowers et al. (2016) found that Chrome Portable temporarily stored its cache files on the hard drive rather than the USB stick. However, when the USB had been removed, these cache files were removed from the hard drive.

Overall, the results from their investigation showed that in both private and portable browsing modes, no cache files were able to be located on the user’s local system (Figure 10 and Figure 11).



*Figure 10 – Artefacts extracted from different browsers running in private mode – (Flowers, Mansour, & Al-Khateeb, 2016)*

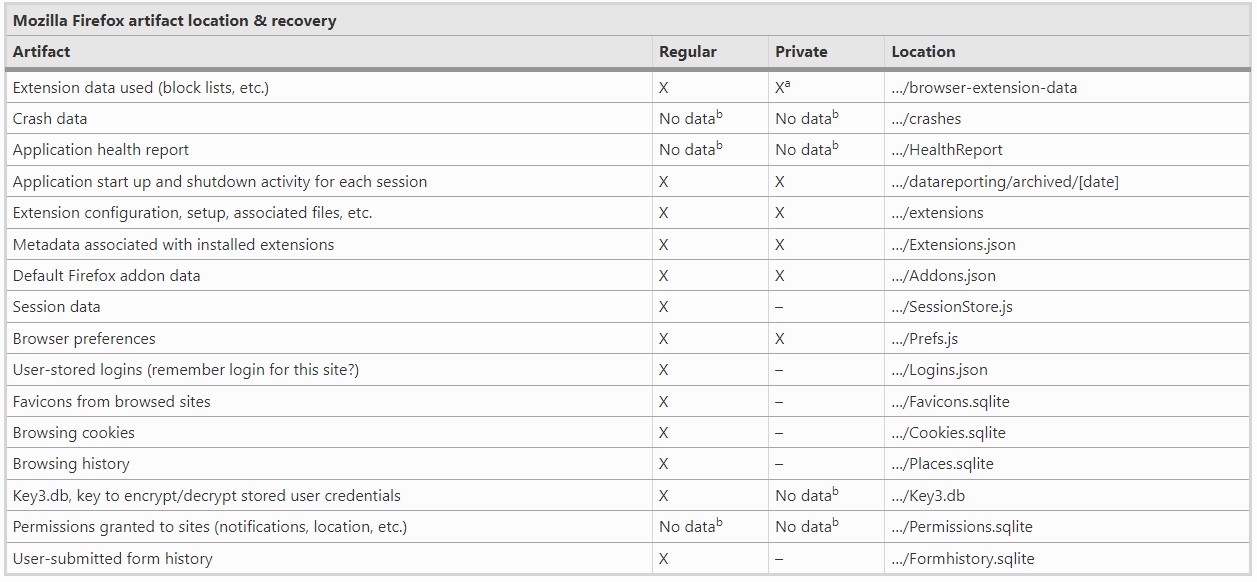


*Figure 11 – Artefacts extracted from different portable browsers running in normal mode – (Flowers, Mansour, & AlKhateeb, 2016)*

#### 2.4.5 Mozilla Firefox – Private Browsing and Firefox Portable

Moving onto Firefox, Nelson et al. (2019) found that the user’s hard drive had more unallocated space after using Private Browsing, compared to the unallocated space after normal browsing. This suggests that files may not have been saved during the private browsing session, or that the files were removed once the session had finished.

Building on these findings, Figure 12 shows the results of the investigation undertaken by Nelson et al. (2019) for Mozilla Firefox.



*Figure 12 – Firefox artefact recovery and location – (Nelson, Shukla, & Smith, 2019)*

Elaborating on the Private Browsing mode offered by Firefox, Flowers et al. (2016) observed that Firefox performed clean-up operations which manipulated Firefox cache files once the private browsing session had finished (Figure 10).

In addition to this, Flowers et al. (2016) also observed the effect Firefox portable had on the availability of cache files on the users’ host system, claiming that there was very little evidence of cache files present (Figure 11).

### 2.5 Web Browser Cache Mechanisms and Structures

Even with the availability of browser cache dependent on whether portable or private browsing modes have been used, browser cache is still a vital forensic artefact that every forensic investigator should aim to acquire and analyse.

As stated by Gupta, Varol, & Zhou (2023) during their investigation into Discord, this acquisition of browser cache can lead to the recreation of web page content, the same way that a suspect may have viewed the content. Therefore, it’s important to explore the ways different browsers cache web page content and understand how this information can be extracted.

#### 2.5.1 Comparison of the Caching Mechanisms for Google Chrome and Mozilla Firefox

Starting with Google Chrome, Hassan (2019) recognises that the architecture of Google Chrome is inherited from Chromium, which is an open-source browser project by Google. Chromium is used by a wide range of modern browsers and implements its own cache mechanism for storing browsing data, known as Disk Cache. This caching mechanism is consistent across all applications that use Chromium as its foundation, and thus, parsing the cache files of different Chromium-based applications should remain consistent.

During their investigation into the Discord application, Iqbal, Motylinski, & MacDermott (2021) confirm this theory by showing that Discord inherits its caching structure from Chromium’s Disk Cache. Therefore, it’s evident that if different applications are using the same underlying caching mechanism, the parsing of the cache files is to be identical.

Moving onto Mozilla Firefox, Hassan (2019) states that Firefox is a completely free and opensource browser that is developed and supported by Mozilla. This is helpful because it offers transparency in its underlying mechanisms and browser functionality. Yet, even with Mozilla

Firefox being completely open source, there’s less academic material on its latest caching mechanism compared to chrome.

As covered in the change logs, Mozilla (2014) released Version 32.0 which introduced a new HTTP caching mechanism (cache2). This delivered improved performance and different caching structures compared to the first version.

With Firefox using its own caching mechanism instead of using Chromium’s Disk Cache like Google Chrome, it’s evident that the cache files of both browsers will need parsing separately, as the cache structures may differ. However, the open-source nature of both browser’s makes this analysis easier, as the source code can be reviewed to find the respective browser’s cache structure.

#### 2.5.2 Comparison of the Cache Structure’s for Google Chrome and Mozilla Firefox

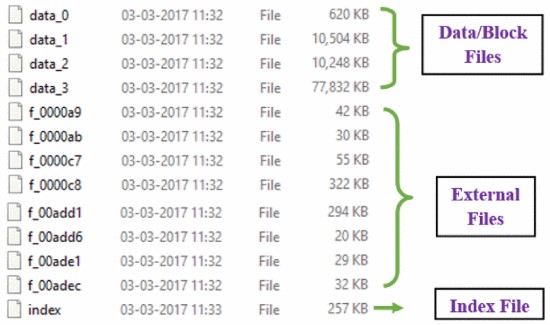
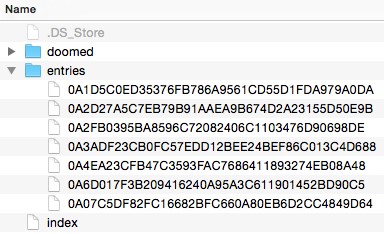
Using existing research to explore the structure of Google Chrome’s cache, Suma, Dija, & Pillai (2017) perform an in-depth analysis of the browser’s cache files. In doing this, they identify the information needed to parse and extract the cache files located on a user’s system.

Furthermore, Firefox cache2 research conducted by Habben (2015) will be used to explore the foundations of Firefox’s cache structure and draw comparisons in the difference between the cache structures of the two web browsers.

##### 2.5.2.1 Cache Folder Comparison

To compare how the cache folders for Google Chrome and Mozilla Firefox differ, Figure 13 shows each of the browser’s cache folders side by side, with Chrome on the left, and Firefox on the right.

Supporting this, Table 3 identifies the similarities and differences between the cache folder’s contents.



*Figure 13 - Google Chrome's cache folder (left) provided by (Suma, Dija, & Pillai, 2017) and Mozilla Firefox's cache folder (right) provided by (Habben, 2015).*

*Table 3 – Table comparing the similarities and differences between the cache folder contents for Google Chrome and Mozilla Firefox - (Habben, 2015; Suma, Dija, & Pillai, 2017)*

|  |  |  |  |
| --- | --- | --- | --- |
| **Similarities** | |  | **Differences** |
| **File Name** | **Description** | **File Name** | **Description** |
| Index File | The index files for Chrome and Firefox have the same role - they point to the cached content.  Chrome’s index file does this through cache addresses that point to cache entries located in one of the four data files.  Whereas Firefox’s index file stores records for each of the entry files located in the “entries” folder.  The index file structure also differs between Chrome and Firefox. | Data Files  External Files | Found in Chrome’s cache folder, these files contain the cache information.  The data files (also known as block files) hold the cached data in structures known as cache entries. Whereas the external files are used to store the cached data if the data exceeds a certain size. |
| Entry Files | Found in Firefox’s cache folder, these files hold all the cached data.  This differs to Chrome because it does not fragment the cached data across multiple files. |

##### 2.5.2.2 Google Chrome’s Cache File Structures

Diving deeper into the structure of the browser’s cache files, Suma et al. (2017) not only outlines each of the file’s individual structure, but also their relationship with each other.

Suma et al. (2017) noted that the index file is split into three parts: the Index header, the Last Recently Used (LRU) data, and the Index Hash Table. The sections of most interest being the index file’s header and hash table, where the hash table contains the cache addresses that point to cache entries located inside one of the four data files.

However, when comparing their findings to the official source code, it’s evident that the research performed by Suma et al. (2017) is outdated. Table 4 takes the index file structure as defined by Suma et al. (2017), and performs a side-by-side comparison using the structures defined in the source code provided by Chromium (2022a).

*Table 4 – Comparison of Chrome’s index file structure - red names indicate fields that weren’t present in the research conducted by Suma et al. (2017), but were included in the official source code provided by Chromium (2022a)*

|  |  |  |  |
| --- | --- | --- | --- |
| **Comparison of Chrome’s Index File Structure** | | | |
| **(Suma, Dija, & Pillai, 2017)** | | (Chromium, 2022a) | |
| **Name** | **Size (Bytes)** | **Name** | **Size (Bytes)** |
| Signature | 4 | magic | 4 |
| Minor Version | 2 | version | 4 |
| Major Version | 2 | num\_entries | 4 |
| Number of Entries | 4 | old\_v2\_num\_bytes | 4 |
| Data Size | 4 | last\_file | 4 |
| Last Created File Number | 4 | this\_id | 4 |
| Unknown | 8 | stats | 4 |
| Table Size | 4 | table\_len | 4 |
| Unknown | 8 | crash | 4 |
| Created Time | 8 | experiment | 4 |
| Padding | 208 | create\_time | 8 |
| LRU | 112 | num\_bytes | 8 |
| Hash Table | ~ | padding | 200 |
|  | | LRU | 112 |
| Hash Table | ~ |

As previously mentioned, cache addresses are stored in the index file’s hash table to point to cache entries located in chrome’s data files. Identified by Suma et al. (2017), the process to parse these cache addresses involves complex calculations which adds additional overhead to the overall parsing process of chrome’s cache files. When compared to Chrome’s source code, the structure of the cache addresses hasn’t changed, and the research conducted on them remains the same (see Appendix A for the latest cache address structure).

Following on from this, the research performed by Suma et al. (2017) on chrome’s data files shows that the data files consist of two sections; a file header, and an array of data blocks which contain cache entries. Again, this research is outdated because it differs to the structure of chrome’s cache entry defined in the official source code. Table 5 performs a side-by-side comparison of chrome’s cache entry structure.

*Table 5 - Comparison of Chrome’s cache entry structure - red names indicate fields that weren’t present in the research conducted by Suma et al. (2017), but were included in the official source code provided by Chromium (2022a)*

|  |  |  |  |
| --- | --- | --- | --- |
| **Comparison of Chrome’s Cache Entry Structure** | | | |
| **(Suma, Dija, & Pillai, 2017)** | | (Chromium, 2022a) | |
| **Name** | **Size (Bytes)** | **Name** | **Size (Bytes)** |
| Hash Number | 4 | hash | 4 |
| Next Cache Address | 4 | next | 4 |
| Cache Entry State | 4 | rankings\_node | 4 |
| Creation Time | 8 | reuse\_count | 4 |
| Key Data Size | 4 | refetch\_count | 4 |
| Long Key Data Cache Address | 4 | state | 4 |
| Data Stream Size Array | 16 | creation\_time | 8 |
| Data Stream Cache Array | 16 | key\_len | 4 |
| Key Data (URL) | ~ | long\_key | 4 |
|  | | data\_size | 16 |
| data\_addr | 16 |
| flags | 4 |
| padding | 16 |
| self\_hash | 4 |
| key | 928 |

Furthermore, Suma et al. (2017) also identified that cached data can be fragmented across multiple cache files. This is proven in the “next” and “long\_key” fields which hold a cache address pointing to the location of its actual cache data. The “Data Stream Cache Array” also contains embedded cache addresses that need to be parsed to retrieve the HTTP response and cached webpage resource data. Ultimately, this additional processing increases the overall complexity of parsing Chrome’s cache files.

##### 2.5.2.3 Mozilla Firefox’s Cache File Structures

Reassuringly, the research conducted by Habben (2015) reveals that Firefox cache2 is much simpler than Chromium’s Disk Cache as it requires less processing to extract and parse the cache files.

Starting with the index file, Habben (2015) identified that the file is split into two parts: the index file’s header and individual records. In comparison to the Firefox source code, the research conducted by Habben (2015) is also outdated and new structures have been defined. Table 6 and Table 7 performs a side-by-side comparison of Firefox’s index file’s structures.

*Table 6 – Comparison of Firefox’s index file’s header structure - red names indicate fields that weren’t present in the research conducted by Habben (2015), but were included in the official source code provided by Searchfox (2022)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Comparison of Firefox’s Index File’s Header Structure** | | | |
|  | **(Habben, 2015)** | | (Searchfox, 2022) | |
| **Name** |  | **Size (Bytes)** | **Name** | **Size (Bytes)** |
| Version |  | 4 | mVersion | 4 |
| Last Modified |  | 4 | mTimeStamp | 4 |
| Dirty Flag |  | 4 | mIsDirty | 4 |
|  |  | | mKBWritten | 4 |

*Table 7 – Comparison of Firefox’s index file’s record structure - red names indicate fields that weren’t present in the research conducted by Habben (2015), but were included in the official source code provided by Searchfox (2022)*

|  |  |  |  |
| --- | --- | --- | --- |
| **Comparison of Firefox’s Index File’s Record Structure** | | | |
| **(Habben, 2015)** | | (Searchfox, 2022) | |
| **Name** | **Size (Bytes)** | **Name** | **Size (Bytes)** |
| Hash of URL | 20 | mHash | 20 |
| Frecency | 4 | mFrecency | 4 |
| Expiration Date | 4 | mOriginAttrsHash | 8 |
| AppID | 4 | mOnStartTime | 2 |
| Flags | 1 | mOnStopTime | 2 |
| File Size | 3 | mContentType | 1 |
|  |  | mFlags | 4 |

Moving onto Firefox’s individual entry files, Habben (2015) suggested that the entry files are again split into two sections: the original cached webpage resource data, and the metadata from the server. To begin processing the metadata, Habben outlined the additional calculations required to find the starting offset of the metadata header. However, when processing the metadata header, the structure identified by Habben differed to the structures defined in Firefox’s source code.

Table 8 takes the metadata header structure defined by Habben (2015), and performs a side-byside comparison using the structure defined in the source code provided by Searchfox (2021).

*Table 8 – Comparison of Firefox’s entry file metadata header structure - red names indicate fields that weren’t present in the research conducted by Habben (2015), but were included in the official source code provided by Searchfox*

*(2021)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Comparison of Firefox’s Entry File Metadata Header Structure** | | | |
|  | **(Habben, 2015)** | | (Searchfox, 2021) | |
| **Name** |  | **Size (Bytes)** | **Name** | **Size (Bytes)** |
| Version |  | 4 | mVersion | 4 |
| Fetch Count | | 4 | mFetchCount | 4 |
| Last Fetched Data | | 4 | mLastFetched | 4 |
| Last Modified Date | | 4 | mLastModified | 4 |
| Frecency | | 4 | mFrecency | 4 |
| Expiration Date | | 4 | mExpirationTime | 4 |
| Key Length | | 4 | mKeySize | 4 |
| URI | | ~ | mFlags | 4 |
|  | |  | URI | ~ |

Lastly, the metadata elements come directly after the metadata header. As mentioned by Habben (2015), these metadata elements are delimited by null bytes and provide key information of interest, such as the HTTP request and response data.

##### 2.5.2.4 Summary

To conclude the cache structures of Chrome and Firefox, the cache structures of Firefox cache2 seems to be less complex when compared to Chromium’s Disk Cache. In Firefox, information relating to the cached content is stored in its own entry file. Whereas in Chrome, this information could be spread across multiple data files or external files and requires cache addresses to be processed before locating the cached data.

#### 2.5.3 Cache Parsers for Google Chrome and Mozilla Firefox

When it comes to the parsing of disk cache using the analysis performed by Suma et al. (2017), a range of tools have been built.

ChromeCacheView is a tool developed by NirSoft (2022a), and is used to view and extract the stored local cache for Google Chrome. Alongside this tool, NirSoft (2022b) developed MZCacheView, which is a tool that views and extracts the local cache folder for Mozilla Firefox on Windows.

These two tools are arguably the most recognised within the forensic community, with Horsman

(2018b) mentioning ChromeCacheView’s importance in facilitating the parsing and extraction of Chrome cache folders during their investigation into the reconstruction of live-streamed video content. The advantage of using these tools is that they are continuously being developed, with the latest versions of each being released in 2022 and come with the ability to extract the cache files of Google Chrome and Mozilla Firefox on demand.

Iqbal, Motyliński, & MacDermott (2021) support Horsman (2018b) by claiming that

ChromeCacheView is currently the best solution available for the forensic analysis of Chromium

Disk Cache. However, Iqbal, Motyliński, & MacDermott (2021) also identify the tool’s limitations during their experimentation, which Horsman (2018b) does not do. These limitations include an incomplete extraction of cached webpage resources, alongside an apparent inability to produce a report on the recovered data.

### 2.6 Summary

To summarise, this thematic literature review provided an insight into the different branches of forensics and critically reviewed literature surrounding HTTP, HTTP Caching Mechanisms, and Portable and Private Web Browsing Capabilities. Additionally, this review explored the cache mechanisms of two different windows-based browsers: Google Chrome and Mozilla Firefox. It was found that HTTP caching store resources such as images, videos, metadata, and URLs which can be extracted during digital forensics to uncover the user’s browsing activities. However, the use of portable and private web browsing capabilities directly impacts the availability of browser cache, further presenting challenges when extracting the cache.

When developing the cross-browser cache parser, theories and models from the reviewed literature will be taken forward into the methodology to help inform the project’s experiment methods. These theories and models include the cache mechanisms of Google Chrome and Mozilla Firefox, their individual cache structures so that all relevant cache information can be extracted, and the relationships between the browser’s cache files so that each file can be parsed.

Lastly, the MVP Cross-Browser Cache Parser, ChromeCacheView, and MZCacheView are evaluated to determine the performance and capabilities of each cache parser. This evaluation will aim to test the limitations previously identified in the literature review, such as testing the number of cached webpage resources recovered, the amount of cache information extracted, and any reporting capabilities present.

## 3. Methodology

### 3.1 Introduction

To contribute towards the project’s aim of developing an MVP cross-browser cache parser for the latest, windows-based versions of Google Chrome and Mozilla Firefox, literature has been critically analysed to explore the browser’s cache structures and inform the methods.

This methodology will set out the methods that will need to be applied to achieve this aim, while also considering any quality issues by preserving reliability and validity, minimising error, and bias, and addressing any ethical considerations

The literature review methods outline the processes which have been followed to select the sources used within the review. Project management methods are then discussed, where separate management methods are identified for the development of the cache parser. Afterwards, the experiment method defines the requirements of the cache parser, alongside processes that will be used to develop the cache parser’s functionality and create the data used in the evaluation. Finally, the evaluation methods then outline the processes that will be taken to evaluate the cache parser’s performance and capabilities.

### 3.2 Quality Issues

It is important that throughout the project, reliability and validity are preserved, bias and error are minimised, and ethical considerations are addressed.

#### 3.2.1 Reliability

Reliability will be preserved by ensuring that steps are taken to allow others to repeat the experiment and achieve the same results. These steps include documenting and displaying the processes that will be followed for the development of the MVP cache parser, the generation of the test data, and the evaluation of the cache parsers.

Moreover, replicating the data generated in the experiment is infeasible due to changing webpage content and adverts. Therefore, any conclusions made using the test data will keep this in mind, and the test data can be made available on request.

More details on the actions that are to be taken to consider reliability in each of the methods, can be found in Appendix B.

#### 3.2.2 Validity

Furthermore, validity will be preserved by ensuring that the evaluation metrics and results accurately reflect the project’s aim of developing an MVP cross-browser cache parser. This will be achieved by selecting metrics which accurately evaluate and reflect, both the performance and capabilities of the cache parsers, qualitatively and quantitatively.

#### 3.2.3 Error

To minimise the risk of error, a variety of error handling practices must be incorporated into the cache parser to ensure continuous execution if unexpected situations arise. This includes guaranteeing that the parser can handle invalid user inputs from the command line.

Further details on the actions taken to minimise the risk of error can be found in Appendix C.

#### 3.2.4 Bias

Due to ethical and privacy concerns, real-world data cannot be used during the evaluation of the cache parser. Therefore, the data will need to be self-generated which causes bias in the data as it may not be representative of the cache files found in real-world scenarios. To reduce the impact of this bias, the data generation process will be kept in mind when drawing conclusions, and the process taken to generate the test data will be documented to allow a better understanding of the results relative to the self-generated data.

More details on the actions that will be taken to account for bias can be found in Appendix D.

#### 3.2.5 Ethical Considerations

Finally, a variety of steps to address ethical concerns related to the project have been identified. As previously explained, the parser will be tested on self-generated data. Since the cache parser is an MVP, and there are no plans to publicly release the parser, the use of self-generated data creates no major ethical concerns. However, if the project is to be developed and taken further to the point of publishing, then significant ethical and legal issues around user privacy need to be addressed. This should involve consulting the appropriate legal statutes (Computer Misuse Act 1990, Data Protection Act 2018, and Investigatory Powers Acts 2016) and checking the legal status of the parser with the appropriate legal teams.

### 3.3 Literature Review Methods

The thematic literature review was carried out with the aim of discovering literature which provided the reader with a clear understanding of browser cache concepts, which are also fundamental to the cache parsers development.

The review was carried out following the summarised process outlined below:

1. Reputable repositories, listed in Appendix E, were searched for using the keywords listed in Appendix F to refine the search results to specific topics of interest.
2. In most cases, these results were then filtered further to restrict the results to recently published work (2018 onwards).
3. The refined and filtered results were then analysed and triaged by topic relevance and validity.

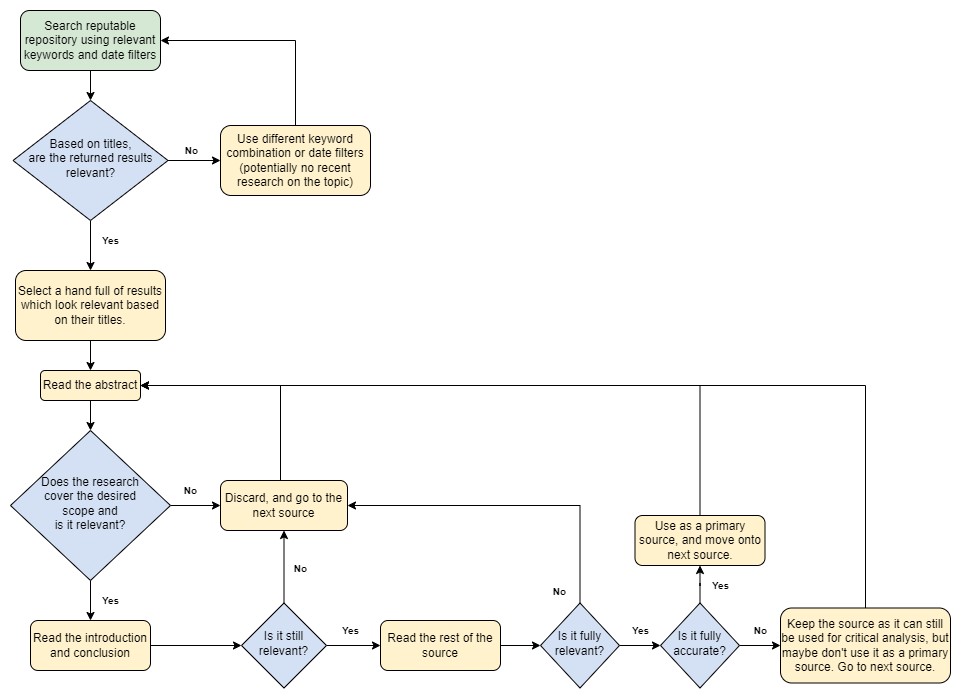
Figure 14 outlines the process which was used to analyse and select sources in greater detail.

*Figure*

*14*

*–*

*Literature review source selection process.*



*Figure created by author using (diagrams.net, 2023).*

### 3.4 Project Management Methods

For the management of the overall project, a Gantt chart has been adopted due to its projectspecific, linear structure, shown in Appendix G. As described by Dori & Sharon (2017), a Gantt chart is a graphical model that allows for the planning, co-ordination, and tracking of projects. This is ideal for this project as it allows for the scheduling of tasks that are required to achieve the project’s objectives defined in Section 1.4.

For the development of the cross-browser cache parser specifically, there are a variety of project management methods that could be adopted. Based on the comparison of prioritisation techniques performed by AlexSoft (2019), the MoSCoW framework has been identified and selected as the chosen project management method.

MoSCoW is part of the DSDM agile method defined by the Agile Business Consortium (2014) and is described as a prioritisation technique to help understand and manage project priorities. It will help to prioritise the cross-browser cache parser’s requirements by splitting each requirement into separate categories, which are detailed further in Appendix H.

### 3.5 Experiment Method

#### 3.5.1 Introduction

To achieve the aim of the project, which is to develop an MVP cross-browser cache parser, the process summarised below will be followed. This process will incorporate the previously identified quality issues, the aims and objectives of the project, and the MoSCoW requirements that will be identified for the development of the cache parser.

1. Identify the cache parser’s MoSCoW requirements.
2. Define processes that will be used for the development of the cache parser.
3. Generate the test data that will be used to evaluate the performance of the MVP and other chosen cache parsers.

#### 3.5.2 Cache Parser MoSCoW Requirements

Combining the MoSCoW prioritisation technique and the limitations of publicly available cache parsers identified in the literature review, Table 9 defines the categorised requirements for the MVP cross-browser cache parser.

*Table 9 – Cache Parser MoSCoW Requirements*

|  |  |
| --- | --- |
| **Category** | **Requirement** |
| Must Have | Multi-browser Cache Identification and Parsing Mechanisms |
| Cache Recovery Mechanism |
| Should Have | CLI Integration |
| Reporting Mechanism |
| Could Have | Modular, Accessible, and Portable Code |
| Error Detection and Handling |
| Won’t Have | Additional Browser Support |

#### 3.5.3 Cache Parser Development

To develop the MVP cross-browser cache parser, the Python programming language will be used alongside the libraries defined in Appendix I. Python has been selected because of previous experience developing with the language, alongside its extensive community and compatibility with other applications.

For each of the MoSCoW requirements defined above, a brief explanation on how each requirement will be implemented in the cache parser’s code will be provided, alongside the specific processes that will be followed defined in the appendices.

##### 3.5.3.1 (Must Have) – Multi-Browser Cache Identification and Parsing Mechanism

Firstly, the cache parser must be able to identify the presence of Google Chrome and Mozilla Firefox on a system, using the following simplified process:

1. Recursively iterate the inputted cache directory.
2. Use a regular expression to identify the cache folders of Chrome and Firefox.

For more details on the full process and regex used, see Appendix J.

The parser must then process and parse each of the cache files for Chrome and Firefox to extract all the cache information.

For Chrome, the parser will follow the summarised process defined below:

1. Check if Chrome’s index file exists. If it does:
   1. Open Chrome’s index file and process the cache addresses found in the hash table.
   2. Use cache addresses to locate cache entries.
   3. Extract the cache information held in each cache entry.
   4. Collate the extracted cache information so it’s ready for reporting.
2. Otherwise, log an error.

Moving onto Firefox, the parser will follow the summarised process defined below:

1. Check if Firefox’s index file exists.
   1. If it does exist, then extract cache records from index file and move on.
   2. If it does not exist, then just move on.
2. Iterate through Firefox’s cache entry folder.
3. Parse each entry individually, extracting the stored cache information.
4. Collate the extracted cache information so it’s ready for reporting.

More details on the processes that will be taken by the cache parser to parse Chrome’s cache files can be found in Appendix K, and Appendix L for Firefox’s cache files.

##### 3.5.3.2 (Must Have) – Cache Recovery Mechanism

Once all cache files have been parsed, the cache parser must then be able to recover and recreate the original webpage resources that were cached by the browser.

To do this, the cache parser needs to be able to extract the raw bytes from the cache file and determine the method used to compress the data. The parser must then identify the extension type for the cached webpage resource and output it to the filesystem with a unique filename.

For more details on the full process that the cache parser will follow, see Appendix M.

##### 3.5.3.3 (Should Have) – CLI Integration

After the “must have” requirements are complete, the parser should then be improved to prompt and handle any command line arguments inputted by the user. This would significantly improve the accessibility and usability of the cache parser.

To do this, the parser should make use of the “argparse” Python library to initialise a range of arguments which the user can use to customise the execution of the parser. Then, the parser should retrieve the user inputted values and perform verification checks to ensure that the inputs remain valid.

More details on the process the cache parser will take to integrate CLI can be found in Appendix N.

##### 3.5.3.4 (Should Have) – Reporting Mechanism

Alongside CLI integration, the cache parser should be able to collate all information extracted from the browser’s cache files and format it appropriately so that it can be reported on. This mechanism would significantly improve the usability by allowing users to select their preferred report formats.

The cache parser will follow the simplified process defined below:

1. Retrieve the report formats specified by user via command line.
2. Collect all extracted information from the browser’s cache files.
3. Output all information to the specified report formats.

See Appendix O for more details on the full process that the cache parser will use to implement the reporting mechanism.

##### 3.5.3.5 (Could Have) – Modular, Accessible, and Portable Code

With the basic functionality implemented, improving the code to be modular and accessible would help the scalability of the program by allowing future modules to be added with ease.

To develop modular code, object orientation (OO) will be used to structure the code into separate classes where specific functionality can be re-used between the parsing of Chrome and Firefox. See Appendix P for more details regarding the structure of the cache parsers code.

Moreover, version control will be implemented by storing the code on GitLab to allow code changes to be documented as the cache parser evolves. Similarly, Google’s style guide for python will be adopted to further manage the accessibility of the code by ensuring consistency across all code components.

Finally, the complete MVP cache parser will use “PyInstaller” to convert the code into a single executable file so that all libraries and code components are bundled into one, improving the overall portability of the cache parser.

##### 3.5.3.6 (Could Have) – Error Detection and Handling

Alongside code styling, appropriately handling any errors would increase the parser’s resilience and improve the user’s overall experience. Therefore, common practices such as exception handling and logging will be used to catch and report on errors, with built-in tests used to catch anomalies.

##### 3.5.3.7 (Won’t Have) – Additional Browser Support

Finally, due to a lack of resources, time constraints, and the nature of MVP, adding support for additional browsers such as Brave or Safari won’t be implemented. However, this does not mean these can’t be visited later for future work.

#### 3.5.4 Data Generation

Once the development is complete, the process behind generating the test data used for the evaluation can begin and will follow the processes defined below.

##### 3.5.4.1 Environment Configuration Process

To generate the test data, two virtual environments are created. The first environment is used to generate the test data, and the second is used to download the standard installers for the browsers.

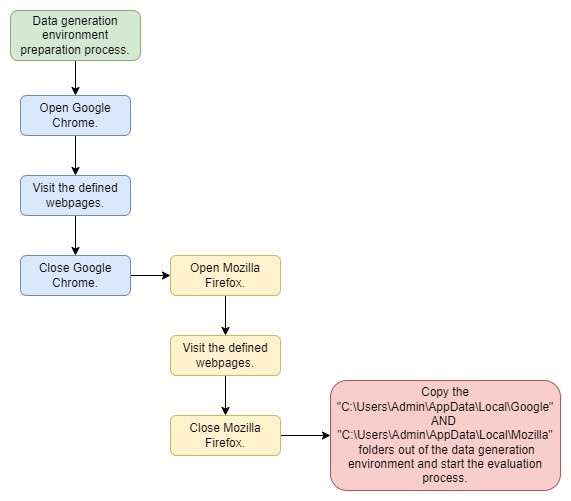
Separate virtual environments are used to prevent contaminating the data generation environment with cache files created when downloading the browser’s standard installers. Additionally, the installers are only downloaded from the official Firefox and Chrome sources to avoid downloading potentially malicious versions.

For more details on the environment set-up process and the environment specifications, see Appendix Q.

Once the standard installers for the browsers have been downloaded, they can be transferred onto the data evaluation environment and installed using the standard installation process, defined in Appendix R.

##### 3.5.4.2 Data Generation Process

With the data generation environment created, Figure 15 defines the process that will be taken when generating the test data for the evaluation.



*Figure 15 – Flowchart defining the process taken to generate the cache files used during the evaluation of the cache parser tools.*

*Figure created by author using (diagrams.net, 2023).*

Supporting this, different webpages have been selected based on the variety of webpage content which would get cached upon visiting the webpage. For more details on the specific webpages selected, see Appendix S.

### 3.6 Evaluation Methods

Using the test-data generated, the performance and capabilities of the MVP cross-browser cache parser will be evaluated against two publicly available cache parsers.

This section will outline the process that will be taken to quantitatively evaluate the performance of the cache parsers using performance metrics, alongside questions that will be used to qualitatively evaluate their capabilities.

#### 3.6.1 Selected Cache Parsers

To assist in the evaluation of the MVP cache parser, the evaluation of two other parsers created will also be performed so that their capabilities and performance can be compared.

Table 10 defines the two publicly available cache parsers selected for the evaluation.

*Table 10 – Publicly available cache parsers selected for the evaluation.*

|  |  |  |
| --- | --- | --- |
| **Application** |  | **Version** |
| ChromeCacheView | 2.41 |  |
| MZCacheView | 2.21 |  |

#### 3.6.2 Performance Metrics Collected

##### 3.6.2.1 Quantitative Evaluation Process

For the quantitative evaluation of the cache parsers, the same execution mode will be used for each of the performance metrics. This execution mode tests the cache parser’s ability to identify browser cache files, extract cache information from the browser cache, and recover the cached webpage resources. Subsequently, outputting the recovered resources and an XML report containing the extracted cache information to the filesystem. In addition to this, the cache folders of either Chrome or Firefox will be directly passed to the parsers.

For more details on the commands and paths which will be used, see Appendix T.

Using built-in windows applications, a range of performance metrics will be collected from the parser’s during their execution. Each of the metrics have been carefully selected to evaluate the performance of the cache parsers based on their output and effect on the system. The metrics can be seen listed below:

* Total number of cached webpage resources identified and recovered.
* Percentage of processor time used by the process.
* Amount of memory consumed by the process.
* Total IO data operations per second for the process.

For more information on the specific metrics collected during the evaluation, see Appendix U.

##### 3.6.2.2 Qualitative Evaluation Process

Moving onto the qualitative evaluation, a comparison will be performed based on the cache parser’s features and the cache information which is extracted from the browser’s cache. This is important because even if one parser is less performant than the other, it may still include capabilities which the other does not.

For the full list of questions that will be used during the qualitative evaluation, see Appendix V.

#### 3.6.3 Evaluation Process

For the evaluation process itself, a separate evaluation environment will be created using the process defined in Appendix Q.

The summarised evaluation process involves:

* Transferring the test-data (cache files) onto the VM.
* Collecting all quantitative performance metrics using the tools specified.
* Performing the comparison for the qualitative evaluation.

The process to collect the quantitative performance metrics using performance monitor will be repeated 3 times so that averages of each metric can be taken. This is to account for anomalies and improve the reliability of the results.

More details on the full evaluation process that will be followed can be found in Appendix W, alongside more information on how windows performance monitor has been used during the evaluation in Appendix X.

### 3.7 Summary

To summarise, this methodology has covered the literature review, project management, experiment, and evaluation methods while also considering any quality issues present. Supporting this further, the processes which will be used have been visualised in the forms of flowcharts, and additional details have been included in the appendices where needed.

## 4. Results and Discussion

### 4.1 Introduction

To assist in achieving the project’s aim and objectives, this section presents the artefacts and results produced by this project.

Reflecting on the evaluation process outlined in the methodology, any changes made during the evaluation will be addressed in the changes section, including any limitations these changes have on the results.

Furthermore, the development of the MVP cross-browser cache parser was vital for this evaluation process. Combining the research conducted in the literature review, and the experiment methods of the methodology, the technical artefacts section provides details on the developed MVP cross-browser cache parser.

Finally, the quantitative and qualitative results gathered from the evaluation of the cache parsers are then displayed and appropriately analysed in their respective discussions.

### 4.2 Changes

When generating the results, ChromeCacheView and MZCacheView couldn’t output both the cached webpage resources and report at the same time. Therefore, the commands defined in Appendix T were changed to generate the XML reports separately for these cache parsers. On the basis that these cache parsers will only output one less file than the MVP cache parser, it’s unlikely this change will have a significant impact on the results. However, it’s still important to note that the performance metrics collected for ChromeCacheView and MZCacheView won’t include the generation of a report.

In addition to this, the cache folder path for MZCacheView was changed because it needed the specific Firefox cache entry folder. Fortunately, this didn’t limit the results because the MVP crossbrowser cache parser was able to handle the new cache folder path.

For more details on the changes made during the evaluation, see Appendix Y.

### 4.3 Technical Artefact – MVP Cross-Browser Cache Parser

Using the processes defined in the experiment methods, a fully functional MVP cross-browser cache parser has been developed using the Python programming language and includes built-in Chrome and Firefox cache parsing functionality. To support this section, the full code for the MVP cross-browser cache parser can be made available upon request.

The MVP cache parser is a command-line tool that allows the user to control its capabilities by passing it a variety of arguments, which also contains built-in documentation and will handle any invalid inputs appropriately. The CLI Integration process shown in Appendix N was used to implement this functionality, and further examples of the cache parser’s command-line integration can be found in Appendix Z.

Upon execution, the cache parser implements the multi-browser cache identification process shown in Appendix J to identify the cache folders for Google Chrome and Mozilla Firefox. Appendix AA shows the code behind this process. Once a cache folder has been identified, the parser uses the processes defined in Appendix K and Appendix L to parse and extract the contents of individual cache files.

For Chrome, the cache parser relies on parsing the index file first to identify and point to cache entries present in the data and external cache files. However, for Firefox, the cache parser can individually parse Firefox’s cache entry files without an index file being present. For code snippets of these parsing mechanisms, see Appendix AB for Chrome and Appendix AC for Firefox.

Embedded inside these parsing mechanisms, cache structures are used to extract the information from the cache files. These structures are constructed from the analysis performed in the literature review, where the research conducted by Suma et al. (2017) and Habben (2015) on Chrome and Firefox’s cache structures was compared to the structures defined in the browser’s source code. For an example of how these cache structures are defined in the parser’s code, see Appendix AD.

The user can also choose to recover the cached webpage resources which are embedded inside the browser’s cache files. When selected, the parser uses the process defined in Appendix M to output the decompressed bytes of the webpage resources. Snippets of code for the overall recovery can be found in Appendix AE.

When execution is complete, the cache parser outputs a report and log file. The report contains all the extracted cache information so that the user can independently process and analyse the cache information. These reports come in a variety of formats and are created using the process defined in Appendix O. More detail on the parser’s reporting mechanism can be found in Appendix AF. Alongside this report, the log file is part of the parser’s error detection and handling functionality, which contains status messages outputted during execution. Examples of the error handling and log files can be seen in Appendix AG.

To conclude, the final version of the MVP cache parser integrates all requirements and processes previously defined using the MoSCoW prioritisation framework. From the development process, it was also clear that Chrome and Firefox have very different caching mechanisms which needed to be parsed separately.

### 4.4 Quantitative Evaluation

Starting with the quantitative evaluation, these results reflect the performance of the cache parsers.

#### 4.4.1 Quantitative Evaluation Results

Table 11 shows the total number of cached webpage resources identified and recovered when using the MVP cross-browser cache parser, ChromeCacheView, and MZCacheView to parse Chrome and Firefox cache files.

*Table 11 – Quantitative evaluation results showing the total number of cached webpage resources identified and recovered by the MVP cross-browser cache parser, ChromeCacheView, and MZCacheView.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Total Number of Cached**  **Webpage**  **Resources** | **Chrome Results** | |  | **Firefox Results** | |
| **MVP Cross-**  **Browser**  **Cache Parser**  **(Chrome)** | **ChromeCacheView** | **MVP Cross-**  **Browser**  **Cache Parser (Firefox)** | **MZCacheView** |
| **Identified** | 522 | 522 | 2108 | 2108 |
| **Recovered** | 501 | 497 | 2064 | 2108 |

Table 12 shows the performance metrics collected from the MVP cross-browser cache parser and ChromeCacheView during the parsing of Chrome cache files (see Appendix AH for the full results)

*Table 12 – Quantitative evaluation results showing the performance metrics collected after using the MVP CrossBrowser Cache Parser and ChromeCacheView to parse chrome cache files.*

|  |  |  |
| --- | --- | --- |
|  | **MVP Cross-Browser**  **Cache Parser**  **(Chrome)** | **ChromeCacheView** |
| **Average percentage of processor time used**  **(%)** | 89.17 | 28.50 |
| **Average amount memory consumed**  **(MB)** | 108.66 | 3.34 |
| **Average Number of**  **IO Read/Write**  **Operations per second** | 395.61 | 278.56 |

Table 13 shows the performance metrics collected from the MVP cross-browser cache parser and MZCacheView during the parsing of Firefox cache files (see Appendix AH for the full results).

*Table 13 – Quantitative evaluation results showing the performance metrics collected after using the MVP CrossBrowser Cache Parser and MZCacheView to parse firefox cache files.*

|  |  |  |
| --- | --- | --- |
|  | **MVP Cross-Browser Cache Parser**  **(Firefox)** | **MZCacheView** |
| **Average percentage of processor time used**  **(%)** | 89.94 | 44.47 |
| **Average amount memory consumed**  **(MB)** | 243.16 | 3.72 |
| **Average Number of**  **IO Read/Write**  **Operations per second** | 643.50 | 769.16 |

#### 4.4.2 Quantitative Evaluation Discussion

##### 4.4.2.1 The Identification and Recovery of Cached Webpage Resources

Starting with the results shown in Table 11, it is evident that each of the cache parsers are equally effective at identifying cached webpage resources from Chrome and Firefox cache files. However, these results also show that the MVP cross-browser cache parser is more effective at recovering cached webpage resources when parsing Chrome cache files, recovering 4 more resources than ChromeCacheView. Whereas MZCacheView is more effective when recovering cached webpage resources from Firefox cache files, recovering 44 more resources than the MVP cache parser.

Iqbal, Motyliński, & MacDermott (2021) and Horsman (2018b) both stated the importance of parsing cache files during a forensic investigation. However, Iqbal, Motyliński, & MacDermott (2021) also identified that ChromeCacheView had its limitations, discovering that it performs an incomplete recovery of cached webpage resources. The results in Table 11 both support and confirm this limitation as the MVP cache parser was able to recover more cached webpage resources than ChromeCacheView.

##### 4.4.2.2 The Performance Metrics

In general, the results shown in Table 12 and Table 13 show that the MVP cache parser is less efficient than ChromeCacheView and MZCacheView as it consumes more system resources.

For the average percentage of processor time used, the MVP cache parser consumed 60.61% more average processor time than ChromeCacheView, and 45.47% more average processor time than MZCacheView.

Similarly, this trend continues for the memory consumed by the MVP cache parser. Consuming an average of 105.32 more megabytes of memory than ChromeCacheView, and 239.44 more megabytes of memory than MZCacheView.

This large difference in performance is likely due to the MVP cache parser not being optimised for performance. Rather, it has been built to function as an MVP and challenge the capabilities of the publicly available cache parsers. Nevertheless, this suggests that the MVP cache parser would likely parse cache files slower than either ChromeCacheView or MZCacheView.

When analysing the average number of IO read/write operations per second, the efficiency of the cache parsers fluctuated. For Firefox, the results show that the MVP cache parser was more efficient than MZCacheView as it used 125.66 less average IO read/write operations per second. Whereas for Chrome, the MVP cache parser was less efficient than ChromeCacheView as it used 117.05 more average IO read/write operations per second.

This fluctuation in performance is likely due to the number of resources recovered by the MVP cache parser. Since the MVP cache parser recovered less resources than MZCacheView, it performed less IO read/write operations per second. As the MVP cache parser recovered more resources than ChromeCacheView, it performed more IO read/write operations per second. Additionally, despite the results for ChromeCacheView and MZCacheView not including the generation of a report, the results show this didn’t have a significant impact.

### 4.5 Qualitative Evaluation

Moving onto the qualitative evaluation, these results aim to compare the capabilities and features available for each of the cache parsers.

#### 4.5.1 Qualitative Evaluation Results

4.5.1.1 What browsers are supported by the cache parser?

Table 14 shows a list of all the browsers which are supported by the MVP cross-browser cache parser, ChromeCacheView, and MZCacheView.

*Table 14 – List of browsers which are supported by each of the cache parsers.*

|  |  |  |  |
| --- | --- | --- | --- |
|  | **MVP Cross-Browser**  **Cache Parser** | **ChromeCacheView** | **MZCacheView** |
| Chrome | Yes | Yes | No |
| Firefox | Yes | No | Yes |
| Brave | No | Yes | No |
| Opera | No | Yes | No |
| Vivaldi | No | Yes | No |
| Yandex | No | Yes | No |
| Edge | No | Yes | No |
| Brave | No | Yes | No |

4.5.1.2 What report formats do the cache parsers support?

Table 15 shows the different report formats which are supported by each of the cache parsers.

*Table 15 – Comparison of the report file formats supported by the MVP Cross-Browser Cache Parser, MZCacheView, and ChromeCacheView.*

|  |  |  |  |
| --- | --- | --- | --- |
| **File Format** | **MVP Cross-Browser**  **Cache Parser** | **ChromeCacheView** | **MZCacheView** |
| Text File | No | Yes | Yes |
| Tab-delimited Text File | No | Yes | Yes |
| Tabular Text File | No | Yes | Yes |
| HTML File (Horizontal) | No | Yes | Yes |
| HTML File (Vertical) | No | Yes | Yes |
| XML | Yes | Yes | Yes |
| CSV | Yes | Yes | Yes |
| JSON | Yes | No | No |
| XLSX | Yes | No | No |

4.5.1.3 What information is extracted from the browser’s cache files?

Table 16 compares information that is present in Chrome’s cache files to the information extracted and outputted by both ChromeCacheView and the MVP Cross-Browser Cache Parser.

*Table 16 – Comparing the information present in Chrome’s cache files, to the information extracted and outputted by ChromeCacheView and the MVP Cache Parser.*

|  |  |  |
| --- | --- | --- |
| **Chrome Cache**  **Information** | **Present in the output of ChromeCacheView?** | **Present in the output of the MVP Cache Parser?** |
| Key Hash | Yes | Yes |
| Reuse Count | No | Yes |
| Refetch Count | No | Yes |
| Entry Creation Time | No | Yes |
| Key | No | Yes |
| URL | Yes | Yes |
| Website | Yes | Yes |
| Frame | Yes | Yes |
| HTTP Headers | Partial | Yes |

Table 17 compares information that is present in Firefox’s cache files to the information extracted and outputted by both MZCacheView and the MVP Cross-Browser Cache Parser.

*Table 17 – Comparing the information present in Firefox’s cache files, to the information extracted and outputted by MZCacheView and the MVP Cache Parser.*

|  |  |  |  |
| --- | --- | --- | --- |
| **Firefox Cache Information** | | **Present in the output of MZCacheView?** | **Present in the output of the MVP Cache Parser?** |
| Cache File Version | | No | Yes |
| Key | | No | Yes |
| Key Size | | No | Yes |
| URL | | Yes | Yes |
| Fetch Count | | Yes | Yes |
| Last Fetch Time | | Yes | Yes |
| Last Modified Time | | Yes | Yes |
| Expiration Time | | Yes | Yes |
| Frecency | | No | Yes |
| Flags | | No | Yes |
| Common  Metadata  Elements | Security Info | No | Yes |
| Request Method | No | Yes |
| Response Method | No | Yes |
| Response Headers | Partial | Yes |
| CTID | No | Yes |

#### 4.5.2 Qualitative Evaluation Discussion

##### 4.5.2.1 Supported Browser Parsing Capabilities

As shown by the results in Table 14, the MVP Cache Parser supports the parsing of both different cache mechanisms found in Chrome and Firefox. Whereas analysis of the browsers supported by ChromeCacheView reveal that it does not have support for multiple cache mechanisms, with all the supported browsers using Chromium Disk Cache. Similarly, MZCacheView only supports Firefox’s cache mechanism.

##### 4.5.2.2 Supported Report Formats

Developing this further, Table 15 shows that ChromeCacheView and MZCacheView support a wider variety of basic report formats, with a total of 7 supported formats. Whereas the MVP CrossBrowser Cache Parser support less, with only 4 supported formats. Despite this, the MVP cache parser supports the JSON format which neither ChromeCacheView nor MZCacheView does. This JSON format is advantageous because it creates a user-friendly structure of the results which can easily be ingested into database systems like MongoDB for future processing and analysis.

Previously, Iqbal, Motyliński, & MacDermott (2021) claimed that ChromeCacheView lacked the ability to produce reports. However, the results produced from the evaluation prove that ChromeCacheView can provide a report, just not at the same time as recovering cached webpage resources.

##### 4.5.2.3 Information Extracted from the Cache Files

Lastly, it can be observed from the results in Table 16 that the MVP cross-browser cache parser performed a more complete extraction of cache information present in Chrome and Firefox cache files. ChromeCacheView missed 4 pieces of key information present in Chrome’s cache files, while also only performing a partial extraction of the HTTP header. Subsequently, these results support Iqbal, Motyliński, & MacDermott (2021) by showing that ChromeCacheView performs an incomplete extraction.

In addition to this, the trend is again the same for MZCacheView. MZCacheView was unable to extract a large amount of key information present in Firefox’s cache files. This included a lot of the metadata elements which would provide a variety of useful information relating to the cached resources during a forensic investigation.

### 4.6 Results Conclusion

To conclude, this section discussed the MVP cross-browser cache parser and included details on its development. This included how it handles the parsing of Chrome and Firefox cache files considering their different cache mechanisms.

Following this, the critical evaluation and discussion of the performance and capabilities of the MVP cross-browser cache parser highlighted the importance of using different tools during a forensic investigation as it validates the output.

When parsing Chrome cache files, the MVP cross-browser cache parser would be considered the more effective parser due to it recovering more resources and performing a more complete extraction of cache information. Despite consuming more system resources. However, if other Chromium-based web browsers were present on the system, then ChromeCacheView would be recommended over the MVP cross-browser cache parser.

When parsing Firefox cache files, the two evaluated cache parsers should be used interchangeably. MZCacheView should be used to recover the cached webpage resources, with the MVP cache parser then being used to perform the more complete extraction of cache information from the cache files.

As part of these conclusions, it’s important to note that each cache parser was evaluated using the same test data to eliminate the bias caused by using self-generated test data. However, due to changing webpage content, replicating the generation of the test data was deemed to be infeasible. If repeating the evaluation is desired, then the test data can be made available upon request. Additionally, the performance of the cache parsers may differ depending on the test data used.

## 5. Conclusion

### 5.1 Introduction

Following on from the results and discussion, this section summarises any previous conclusions made, as well as reflecting on the project’s aims and objectives by discussing the extent to which they have been achieved. Any contributions to knowledge and limitations encountered during the project are then summarised. Finally, future work is discussed, and a reflection is performed on personal project performance.

### 5.2 Overview of The Project’s Conclusions

Unfortunately, the literature review identified a lack of academic literature which evaluated the performance and capabilities of publicly available cache parsers. Therefore, this project addressed these issues by using the research conducted on the latest browser cache mechanisms to develop an MVP cache parser, which was then evaluated alongside two publicly available cache parsers. The following conclusions have been made during this process.

Firstly, the investigations performed by Flowers et al. (2016) and Nelson et al. (2019) found that no cache files were present on the user’s system after using portable and private versions of Firefox and Chrome. This allowed for the conclusion that these private and portable capabilities are the key factor that affects the availability of cache files.

During the literature review, it was also found that the cache mechanisms and structures of Google Chrome and Mozilla Firefox were vastly different. Chrome inherits its cache mechanism from Chromium Disk Cache, whereas Firefox implements its own cache mechanism, known as cache2. Exploring these mechanisms also revealed the unique structure of each browser’s cache files. Despite these differences, this research informed the development of the MVP cache parser which supports the parsing of Chrome and Firefox cache files.

Also, the critical evaluation of the cache parsers revealed that the MVP cache parser was more effective at parsing Chrome cache files compared to ChromeCacheView. Thus, recommending the MVP cache parser over ChromeCacheView for parsing Chrome’s cache. This decision was made because ChromeCacheView recovered less cached webpage resources and extracted less information from the cache files than the MVP Cache Parser. Further supporting the

ChromeCacheView extraction limitations identified by Iqbal, Motyliński, & MacDermott (2021).

However, ChromeCacheView is recommended over the MVP cache parser when other Chromium-based web browser such as Opera or Brave are present.

This evaluation also elaborated on the importance of using multiple tools to complement each other’s output. When parsing Firefox cache files, MZCacheView was able to recover more cached webpage resources than the MVP Cache Parser. This was still limited because the reports generated by MZCacheView revealed that the parser performed a partial extraction of information from the cache files. Therefore, the MVP cache parser is recommended to compliment the recovery of resources as it can generate reports which contain all the information present in Firefox’s cache files.

In fact, the changes section identified that ChromeCacheView and MZCacheView were both limited as they failed to generate a report after recovering the cached resources. This meant that these reports needed to be generated separately.

### 5.3 Project Reflection

#### 5.3.1 Aim

The project’s aim was to develop an MVP cross-browser cache parser, focusing on the normal installations of Google Chrome and Mozilla Firefox. This was accomplished by reviewing literature on the cache structures of Chrome and Firefox to develop the processes defined in the experiment methods. The development of the MVP cache parser then applied these defined processes alongside the cache parsers MoSCoW requirements to implement the ability to parse Chrome and Firefox cache files. Subsequently, achieving the project’s aim. Despite this success, the development was limited due to a lack of resources and time constraints. Resulting in the MVP cache parser consuming more system resources than publicly available cache parsers.

#### 5.3.2 Objectives

Objective one required critically reviewing contemporary literature relevant to this project’s area to help inform the methodology. This was achieved during the literature review, where literature on HTTP Concepts, HTTP Caching, Portable and Private Web Browsers, and the caches of Chrome and Firefox were critically analysed. However, aspects of this research were limited due to outdated research. Analysis was then performed to critically compare the outdated research on the cache structures of Chrome and Firefox to their official source code. The latest cache structures were then extracted and used to help inform the methodology.

Objective two required taking the outputs of the literature review and using this to develop an experimental methodology. This was accomplished by taking the research performed in the literature review on the latest cache mechanisms and structures of Chrome and Firefox and using it to develop processes and requirements for the MVP Cross-Browser Cache Parser defined in the project’s experimental methodology.

Objective three required performing the primary research and presenting the data needed to develop the MVP cache parser so that it could parse both Chrome and Firefox caches. Like objective one and two, the primary research was performed during the literature review on the cache mechanism and structures of Chrome and Firefox. The experiment methods then defined and presented the data (i.e., processes and requirements) using this research, which would then be used to develop the MVP cache parser so it could parse Chrome and Firefox cache files.

Objective four was to develop an MVP cache parser to recover cached webpage resources. This has been achieved during the development process, which used the processes and requirements defined in the experiment methods to develop a cache parser which supported the parsing of Chrome and Firefox cache files. The evaluation of this cache parser then revealed that it was able to recover the cached resources for both Chrome and Firefox. However, the performance of this parser was limited due to a lack of resources and time constraints.

Objective five required critically evaluating and discussing the performance and capabilities of the MVP cache parser against publicly available cache parsers. The performance of these cache parsers was critically evaluated during the quantitative evaluation using a variety of carefully selected metrics to determine the effectiveness and efficiency of the cache parser. Afterwards, the parsers capabilities were then critically evaluating using a variety of comparative questions during the qualitative evaluation to identify any limitations present. A variety of conclusions and recommendations were then made based on these discussions and can be seen in Section 4.6.

Objective six was to conclude the findings of the project while reflecting and making recommendations based on these findings. This has been achieved in Section 5.2 of this report, where conclusions drawn from the research conducted in the literature review and the evaluation results have been summarised and recommendations on which cache parsers to use have been made.

#### 5.3.3 Research Questions (RQs)

RQ 1 was to identify the differences in cache structures and mechanisms of Chrome and Firefox. This has been achieved during the literature where it was identified that Chrome used Chromium’s

Disk Cache as it’s cache mechanism, and Firefox used cache2. Further comparison of academic literature against browser source code also revealed the differences in cache structures between the browsers.

RQ 2 required the identification of information which can be extracted from the caches of Chrome and Firefox. This was accomplished during the qualitative evaluation which identified all the key information available in the browser’s cache files and compared it to the information extracted by each of the cache parsers.

Finally, RQ 3 was to identify how publicly available cache parsers differed to the MVP crossbrowser cache parser in terms of capability and performance. This has been achieved through the project’s evaluation, where the performance and capabilities of the selected cache parsers were evaluated using a variety of performance metrics and comparative question. The results were then discussed to identify how the publicly available cache parsers differed to the MVP cross-browser cache parser, and which parsers to use in different scenarios.

### 5.4 Limitations

To recap, there were many project limitations which affected the decisions and processes used during this project.

During the literature review, a lot of the literature surrounding the private and portable web browsing capabilities and the cache structures of Chrome and Firefox was outdated, limiting the information which could be carried forward into the methodology without performing additional analysis.

For the limitations present in the methodology, replicating the data generated for the experiment was deemed infeasible because of changing webpage content and adverts present during the data generation process. In addition to this, the generation of the data was inherently limited by ethical and privacy concerns which prevented the use of real-world data. Therefore, the evaluation used self-generated data which doesn’t accurately reflect the state of cache files in a real-world scenario.

Developing the MVP cache parser was also limited due to a lack of resources and time constraints. With more time and resources, the cache parser’s capability could have been expanded to implement support for parsing different browsers or even mobile applications.

Finally, limitations were also present in the results and conclusions. The results were limited because the performance metrics collected for ChromeCacheView and MZCacheView didn’t include the generation of a report file. In addition to this, the results, conclusions, and recommendations were limited due to the use of self-generated data mentioned previously.

### 5.5 Future Work

Reflecting on the limitations and outputs of this project, this project can be expanded in many ways.

Starting with the limitations, it might be beneficial to repeat the evaluation using a variety of realworld data sets. This would help to accurately reflect the cache parser’s ability to extract cache information in different real-world scenarios and would allow for a more reliable evaluation of the cache parsers.

In conjunction with this, the investigations into private and portable web browsers performed by Nelson, Shukla, & Smith (2019) and Flowers et al. (2016) were found to be outdated. Therefore, their investigations could be revisited to use the latest versions of browsers and expand the evaluation of the cache parsers to identify whether they can recover cache information after a private or portable browsing session.

Lastly, the capabilities of the MVP cross-browser cache parser could be expanded to include support for more privacy-oriented browsers such as Brave, or less researched browsers such as Safari. As seen during the qualitative evaluation, ChromeCacheView already supports an extensive amount of Chromium-based web browsers.

### 5.6 Self-appraisal

By undertaking this project, new knowledge surrounding digital forensics has been gained. However, during this process, a variety of challenges were encountered which assist in highlighting personal strengths and weaknesses.

One of the first challenges encountered was the overall project management. Upon completing the project, this is considered a strength as it reinforced the importance and benefit of the Gantt chart. The Gantt chart ensured the overall project was organised and delivered in time.

Occasionally, the timings set using the Gantt chart were unrealistic. For instance, the methodology took longer than originally planned. However, this was likely due to an initial lack of understanding behind the complexity and overall importance of the methodology for this project.

In addition to this, the development of the MVP cache parser was another personal strength. To enable this development, the research into the cache mechanisms and structures of Chrome and Firefox was vital. This knowledge was then successfully converted into a fully functioning parser which could be used to parse the cache files of both Chrome and Firefox. Despite having very little previous knowledge on the cache mechanisms of Chrome and Firefox.

Moving onto weaknesses, the ethical and privacy concerns around the cache parser should have been integrated and addressed more in the overall project. If this was done, then there’s the potential that real-world test data could have been used rather than self-generated test data. In addition to this, the MVP cache parser might have been able to be shared publicly. However, due to time constraints, this was infeasible.

Despite this, the project has provided the foundational knowledge needed to expand the work further.

## 6. References

Afridi, N. (2021). The Current Status of Forensic Science and its Impact on Administration of Criminal Justice System in Pakistan: An Analytical Study. 1-32. Retrieved 02 15, 2023, from https://papers.ssrn.com/sol3/papers.cfm?abstract\_id=3781586

Agile Business Consortium. (2014). *DSDM Project Framework - Chapter 10: MoSCoW Prioritisation*. Retrieved 02 22, 2023, from Agile Business Consortium: https://www.agilebusiness.org/dsdm-project-framework/moscow-prioririsation.html

AlexSoft. (2019, May 16). *The Most Popular Prioritization Techniques and Methods: MoSCoW, RICE, KANO model, Walking Skeleton, and others*. Retrieved 01 26, 2023, from AlexSoft:

https://www.altexsoft.com/blog/business/most-popular-prioritization-techniques-andmethods-moscow-rice-kano-model-walking-skeleton-and-others/

Berner-Lee, T., Fielding, R., & Frystyk, H. (1996). *RFC 1945 - Hypertext Tranfser Protocol - HTTP/1.0.* Internet Engineering Task Force (IETF). doi:10.17487/RFC1945

Brush, K., Rosencrance, L., & Cobb, M. (2021, September). *cybercrime*. Retrieved 02 15, 2023, from TechTarget: https://www.techtarget.com/searchsecurity/definition/cybercrime

Chromium. (2022a, September 14). *disk\_format.h*. Retrieved 01 11, 2023, from Chromium Code Search: https://source.chromium.org/chromium/chromium/src/+/main:net/disk\_cache/blockfile/dis k\_format.h;bpv=0;bpt=1

Chromium. (2022b, September 14). *addr.h*. Retrieved 02 01, 2023, from Chromium Code Search:

https://source.chromium.org/chromium/chromium/src/+/main:net/disk\_cache/blockfile/ad dr.h;bpv=0 diagrams.net. (2023). *app.diagrams.net*. Retrieved from diagrams.net: https://app.diagrams.net/

Dori, D., & Sharon, A. (2017). Model-Based Project-Product Lifecycle Management and Gantt Chart Models: A Comparative Study. *Systems Engineering, 20*, 1-20. doi:10.1002/sys.21407

El-Tayeb, M., Taha, A., & Fayed, Z. T. (2022). Live-Streamed Video Reconstruction for Web Browser Forensics. *Ingenierie des Systemes d'Information, 21*(1), 61-66.

doi:10.18280/ISI.270107

Fielding, R., Gettys, J., Mogul, J., H, F., & Berners-Lee, T. (1997). *RFC 2068 - Hypertext Transfer Protocol - HTTP/1.1.* Internet Engineering Task Force (IETF). doi:10.17487/RFC2068

Fielding, R., Nottingham, M., & Reschke, J. (2022a). *RFC 9110 - HTTP Semantics.* Internet Engineering Task Force (IETF). doi:10.17487/RFC9110

Fielding, R., Nottingham, M., & Reschke, J. (2022b). *RFC 9111 - HTTP Caching.* Internet Engineering Task Force (IETF). doi:10.17487/RFC9111

Flowers, C., Mansour, A., & Al-Khateeb, H. M. (2016). Web browser artefacts in private and portable modes: A forensic investigation. *International Journal of Electronic Security and Digital Forensics, 8*(2), 99-117. doi:10.1504/IJESDF.2016.075583

Google. (2022). *How private browsing works in Chrome*. Retrieved 12 05, 2022, from Google Chrome Help: https://support.google.com/chrome/answer/7440301

Gupta, K., Varol, C., & Zhou, B. (2023). Digital forensic analysis of discord on google chrome.

*Forensic Science International: Digital Investigation, 44*. doi:https://doi.org/10.1016/j.fsidi.2022.301479

Habben, J. (2015, February 10). *Firefox Cache2 Storage Breakdown*. Retrieved 12 12, 2022, from RSSing: https://forensics618.rssing.com/chan-8498287/article87.html

Hariharan, M., Thakar, A., & Sharma, P. (2022). Forensic Analysis of Private Mode Browsing

Artifacts in Portable Web Browsers Using Memory Forensics. *2022 International Conference on Computing, Communication, Security and Intelligent Systems (IC3SIS)*, (pp. 1-5). doi:10.1109/IC3SIS54991.2022.9885379

Hasan, F.-K., Sondos, K.-M., Hussin, H. J., & Ale, H. J. (2021). Forensic analysis of private browsing mechanisms: Tracing internet activities. *Journal of Forensic Science and Research, 5*(1), 12-19. doi:10.29328/JOURNAL.JFSR.1001022

Hassan, N. A. (2019). Web Browser and E-mail Forensics. In *Digital Forensics Basics: A Practical*

*Guide Using Windows OS* (pp. 247-289). Berkeley, CA: Apress. doi:10.1007/978-1-48423838-7\_8

Horsman, G. (2018a). I didn't see that! An examination of internet browser cache behaviour following website visits. *Digital Investigation, 25*, 105-113. doi:10.1016/J.DIIN.2018.02.006

Horsman, G. (2018b). Reconstructing streamed video content: A case study on YouTube and

Facebook Live stream content in the Chrome web browser cache. *Digital Investigation, 26*, 30-37. doi:10.1016/J.DIIN.2018.04.017

Horsman, G., Findlay, B., Edwick, J., Asquith, A., Swannell, K., Fisher, D., . . . McKain, P. (2019). A forensic examination of web browser privacy-modes. *Forensic Science International: Reports, 1*. doi:https://doi.org/10.1016/j.fsir.2019.100036

Hughes, K., Papadopoulos, P., Pitropakis, N., Smales, A., Ahmad, J., & Buchanan, W. (2021). Browsers' Private Mode: Is It What We Were Promised? *Computers, 10*. doi:10.3390/computers10120165

Iqbal, F., Motyliński, M., & MacDermott, Á. (2021). Discord Server Forensics: Analysis and

Extraction of Digital Evidence. *2021 11th IFIP International Conference on New*

*Technologies, Mobility and Security (NTMS)*, (pp. 1-8). doi:10.1109/NTMS49979.2021.9432654

Jadhav, M. R., & Meshram, B. B. (2018). Web Browser Forensics for Detecting User Activities.

*International Research Journal of Engineering and Technology (IRJET), 5*(7), 273-279. Retrieved 02 15, 2023, from https://www.irjet.net/archives/V5/i7/IRJET-V5I748.pdf

Mozilla. (2014, September). *Firefox Release Notes - 32.0*. Retrieved 12 12, 2022, from Mozilla: https://www.mozilla.org/en-US/firefox/32.0/releasenotes/

Mozilla. (2022, October 28). *Evolution of HTTP*. Retrieved 12 14, 2022, from MDN Web Docs:

https://developer.mozilla.org/en-

US/docs/Web/HTTP/Basics\_of\_HTTP/Evolution\_of\_HTTP

Mozilla. (2022). *Private Browsing - Use Firefox without saving history*. Retrieved 12 05, 2022, from Mozilla Firefox Support: https://support.mozilla.org/en-US/kb/private-browsing-usefirefox-without-history

Narayanan, A. S., Rajkumar, T., & Sobhana, N. V. (2017). Forensic analysis of residual artifacts from private browsing sessions in Linux. *Advances in Intelligent Systems and Computing, 479*, 39-49. doi:10.1007/978-981-10-1708-7\_5/TABLES/1

Nelson, R., Shukla, A., & Smith, C. (2019). Web Browser Forensics in Google Chrome, Mozilla

Firefox, and the Tor Browser Bundle. In X. Zhang, & K.-K. R. Choo (Eds.), *Digital Forensic Education: An Experiential Learning Approach* (pp. 219-241). Springer International Publishing. doi:10.1007/978-3-030-23547-5\_12

Nguyen, H. V., Lo Iacono, L., & Federrath, H. (2019). Mind the cache: Large-scale explorative study of web caching. *Proceedings of the ACM Symposium on Applied Computing*, *F147772*, pp. 2497-2506. doi:10.1145/3297280.3297526

NirSoft. (2022a, December 22). *ChromeCachView v2.41 - Cache viewer for Google Chrome Web browser*. Retrieved 01 11, 2023, from NirSoft: https://www.nirsoft.net/utils/chrome\_cache\_view.html

NirSoft. (2022b, October 22). *MZCacheView v2.21 - View the cache files of Firefox Web browsers*.

Retrieved 01 11, 2023, from NirSoft: https://www.nirsoft.net/utils/mozilla\_cache\_viewer.html

Sadiku, M. N., Tembely, M., & Musa, S. M. (2017). Digital Forensics. *International Journal of Advanced Research in, 7*(4), 274-276. doi:10.23956/ijarcsse/V7I4/01404

Sawicki, J., Zych, P., & Sawicki, B. (2021). Practical studies of local HTTP cache security. *CPEE 2021 - 22nd International Conference "Computational Problems of Electrical Engineering"*, (pp. 1-5). doi:10.1109/CPEE54040.2021.9585249.

Searchfox. (2021, September 27). *CacheFileMetadata.h*. Retrieved 01 11, 2023, from Searchfox: https://searchfox.org/mozilla-central/source/netwerk/cache2/CacheFileMetadata.h

Searchfox. (2022, November 12). *CacheIndex.h*. Retrieved 01 11, 2023, from Searchfox: https://searchfox.org/mozilla-central/source/netwerk/cache2/CacheIndex.h

Shafqat, N. (2016). Forensic Investigation of User's Web Activity on Google Chrome using various Forensic Tools. *IJCSNS International Journal of Computer Science and Network Security, 16*(9), 123-132. Retrieved 11 08, 2022

Smartsheet. (2019, February 20). *Gantt Chart Templates in Excel and Other Tools*. Retrieved 11 07, 2022, from smartsheet: https://www.smartsheet.com/gantt-chart-excel-templates

StatCounter. (2022). *Desktop Browser Market Share Worldwide*. Retrieved from StatCounter -

GlobalStats: https://gs.statcounter.com/browser-market-

share/desktop/worldwide/#monthly-202201-202211

Statista. (2022, July). *Number of internet and social media users worldwide as of July 2022*. Retrieved from Statista: https://www.statista.com/statistics/617136/digital-populationworldwide/

Studiawan, H., Sohel, F., & Payne, C. (2019). A survey on forensic investigation of operating system logs. *Digital Investigation, 29*, 1-20. doi:10.1016/j.diin.2019.02.005

Suma, G. S., Dija, S., & Pillai, A. T. (2017). Forensic Analysis of Google Chrome Cache Files. *2017 IEEE International Conference on Computational Intelligence and Computing Research, ICCIC 2017*, 1-5. doi:10.1109/ICCIC.2017.8524272

Tableau. (n.d.). *Collect Data with Windows Performance Monitor*. Retrieved 03 10, 2023, from Tableau - Tableau Server on Windows Help: https://help.tableau.com/current/server/enus/perf\_collect\_perfmon.htm

## 7. Appendices

### 7.1 Appendix A – Chrome Cache Address Structure

• Go Back (Section 2.5.2.2 – Google Chrome’s Cache File Structures)

Table 18 contains the structure used to parse Chrome’s cache addresses.

*Table 18 – Structure of Chrome’s cache addresses - (Chromium, 2022b)*

|  |  |  |
| --- | --- | --- |
| **File Type** | **Field** | **Bits** |
|  | initialized bit | 1000 0000 0000 0000 0000 0000 0000 0000 |
| file type | 0111 0000 0000 0000 0000 0000 0000 0000 |
| **External File** | file number | 0000 1111 1111 1111 1111 1111 1111 1111 |
| **Block File** | reserved bits | 0000 1100 0000 0000 0000 0000 0000 0000 |
| number of blocks | 0000 0011 0000 0000 0000 0000 0000 0000 |
| file selector | 0000 0000 1111 1111 0000 0000 0000 0000 |
| block number | 0000 0000 0000 0000 1111 1111 1111 1111 |

### 7.2 Appendix B Reliability

Go Back (Section 3.2.1 – Reliability)

*Table 19 – Actions that are to be taken to consider reliability in each of the methods.*

|  |  |
| --- | --- |
| **Section** | **Action** |
| 3.3 - Literature Review | Define the source selection and analysis process. |
| List the reputable source repositories used in the literature review. |
| List the keywords used when searching the reputable source repositories. |
| 3.5.3 - Development | Explicitly outline the cache parser requirements. |
| Define the processing flow of key cache parser functionality. |
| Make the source code of the cross-browser cache parser functionality available upon request. |
| 3.5.4 - Data Generation | Define the process used to generate the environments used to generate the test data, with their specifications. |
| Specify the versions of Google Chrome and Mozilla Firefox used. |
| Define the process used to generate the test data, alongside the specific websites visited. |
| Make the data generated for the evaluation available upon request. |
| 3.6 - Evaluation | List the environment specification used for the evaluation, alongside the versions of the publicly available cache parsers. |
| Outline the different tools and metrics used to evaluate the performance of the cache parsers. |
| Define the comparative questions to be asked during the qualitative evaluation. |
| Define the full evaluation process. |

### 7.3 Appendix C Error

Go Back (Section 3.2.3 – Error)

*Table 20 – Actions that are to be taken to minimise the risk of error.*

|  |  |
| --- | --- |
| **Section** | **Action** |
| 3.5.3 - Development | Implement exception handling in cache parser code. |
| Set default values for the cross-browser cache parser’s command line arguments. |
| Embed tests within the code to catch unexpected results. |

### 7.4 Appendix D – Bias

• Go Back (Section 3.2.4 – Bias)

*Table 21 – Actions that will be taken to account for bias in the self-generated test data.*

|  |  |
| --- | --- |
| **Section** | **Action** |
| 3.5.4 - Data Generation | Make the test-data generated for the evaluation available upon request. |
| 3.6 - Evaluation | Use the same generated data for all cache parsers. |

### 7.5 Appendix E Reputable Source Repositories

Go Back (Section 3.3 – Literature Review Methods)

Table 22 contains a list of reputable source repositories which were used to search for different academic sources.

*Table 22 – Reputable source repositories used during the literature review.*

|  |
| --- |
| **Name** |
| Google Scholar |
| IEEE Xplore |
| ACM |
| Internet Engineering Task Force (IETF) |
| Forensics-focused Blogs (James Habben) |

### 7.6 Appendix F Keywords

Go Back (Section 3.3 – Literature Review Methods) Table 23 contains the list of keywords used to refine the search results.

*Table 23 – Keywords used to search and filter for sources during the literature review.*

|  |
| --- |
| **Keyword** |
| Digital Forensics |
| Browser Forensics |
| Browser Artefacts |
| Google Chrome |
| Mozilla Firefox |
| Browser Cache |
| Cache |
| Cache Structures |
| Cache Mechanisms |
| Hypertext Transfer Protocol (HTTP) |
| HTTP Cache |
| Portable Browser |
| Privacy Features |
| Browser Privacy Modes |
| Cache Files |

### 7.7 Appendix G Project Gantt Chart

Go Back (Section 3.4 – Project Management Methods)

The Gantt chart shown in Figure 16 was taken from a template provided by from (Smartsheet, 2019).

*Figure*

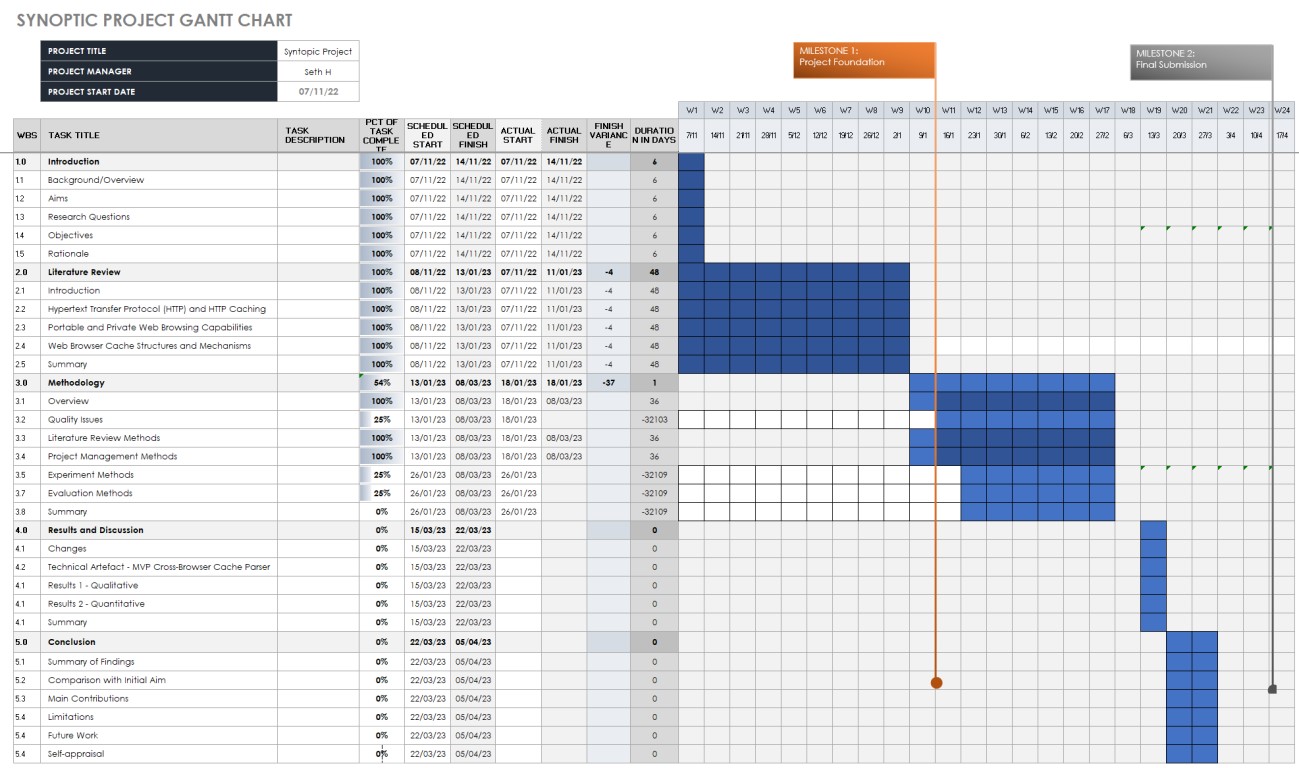
*16*

*–*

*Project Gantt Chart*

*–*

*Methodology Progress*



### 7.8 Appendix H MoSCoW Prioritisation Framework

Go Back (Section 3.4 – Project Management Methods)

Table 24 describes the 4 categories used in the MoSCoW Prioritisation Framework. *Table 24 – Categories of the MoSCoW Prioritisation Framework*

|  |  |
| --- | --- |
| **Category** | **Description** |
| Must Have | Mandatory, non-negotiable requirements. |
| Should Have | Important requirements with substantial impact, but not vital. |
| Could Have | Desirable requirements but have less impact. |
| Won’t Have | Unrealistic requirements not to-be implemented yet. |

### 7.9 Appendix I – Cache Parser Libraries

Go Back (Section 3.5.3 – Cache Parser Development)

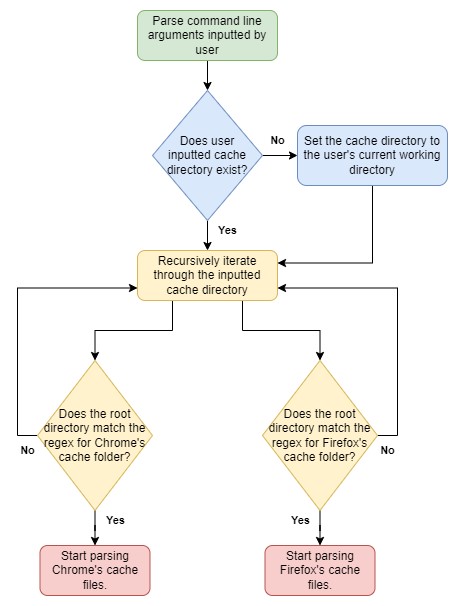
*Table 25 - List of python libraries that will be used for the cross-browser cache parser.*

|  |
| --- |
| **Python Library** |
| argparse |
| binascii |
| brotli |
| csv |
| ctypes |
| datetime |
| dicttoxml |
| gzip |
| hashlib |
| json |
| logging |
| magic |
| mimetypes |
| os |
| pathlib |
| re |
| time |
| typing |
| xlsxwriter |
| xml.dom.minidom |

### 7.10 J – Multi-Browser Cache Identification Process

* 3.5.3.1 – (Must Have) Multi-Browser Cache Identification and Parsing Mechanism)
* Go Back (Section 4.3 – Technical Artefact – MVP Cross-Browser Cache Parser)

Figure 17 outlines cache identification process, with Table 26 displaying the regular expressions used during this process.



*17 Cache Identification Process.*

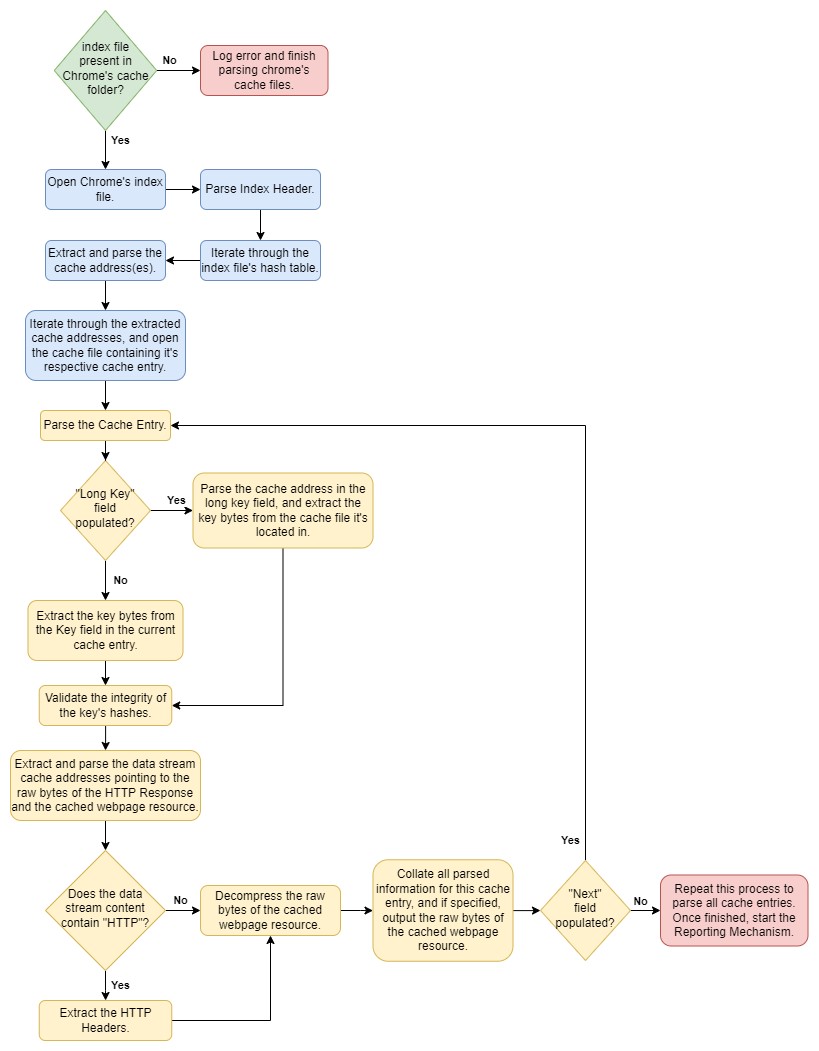
* Go Back (Section 3.5.3.1 – (Must Have) Multi-Browser Cache Identification and Parsing Mechanism)
* Go Back (Section 4.3 – Technical Artefact – MVP Cross-Browser Cache Parser)

*Table 26 – Regular Expressions that will be used to identify cache folders for Chrome and Firefox*

|  |  |
| --- | --- |
| **Browser** | **Regex** |
| Google Chrome | .\*\\Google\\Chrome\\User Data\\\w+\\Cache\\Cache\_Data$ |
| Mozilla Firefox | .\*\\Mozilla\\Firefox\\Profiles\\.\*-release\\cache2(\\entries)?$ |

### 7.11 K – Chrome Cache Parsing Process

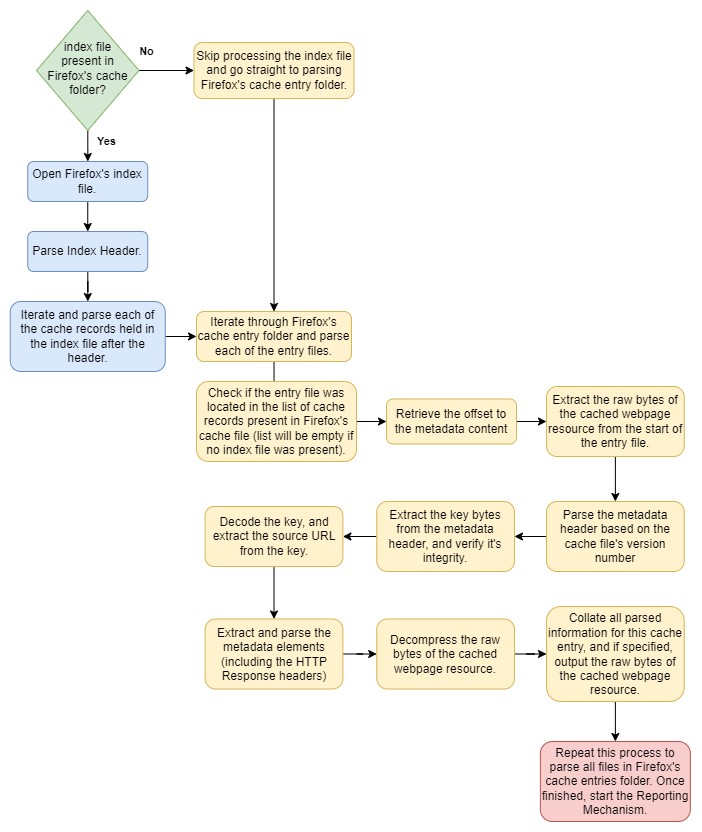
* 3.5.3.1 – (Must Have) Multi-Browser Cache Identification and Parsing Mechanism)
* Go Back (Section 4.3 – Technical Artefact – MVP Cross-Browser Cache Parser)



*18 The logic and processing flow behind the Chrome Cache Parser*

### 7.12 L Firefox Cache Parsing Process

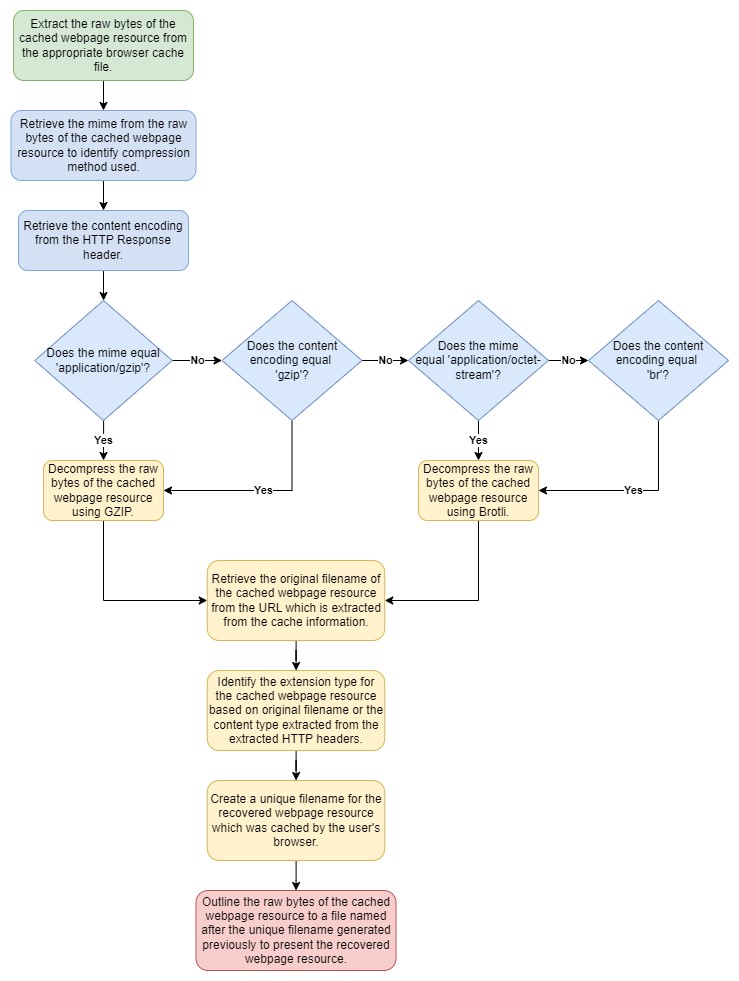
* 3.5.3.1 – (Must Have) Multi-Browser Cache Identification and Parsing Mechanism)
* Go Back (Section 4.3 – Technical Artefact – MVP Cross-Browser Cache Parser)



*19 The logic and processing flow behind the Firefox Cache Parser*

### 7.13 M – Cache Recovery Mechanism Process

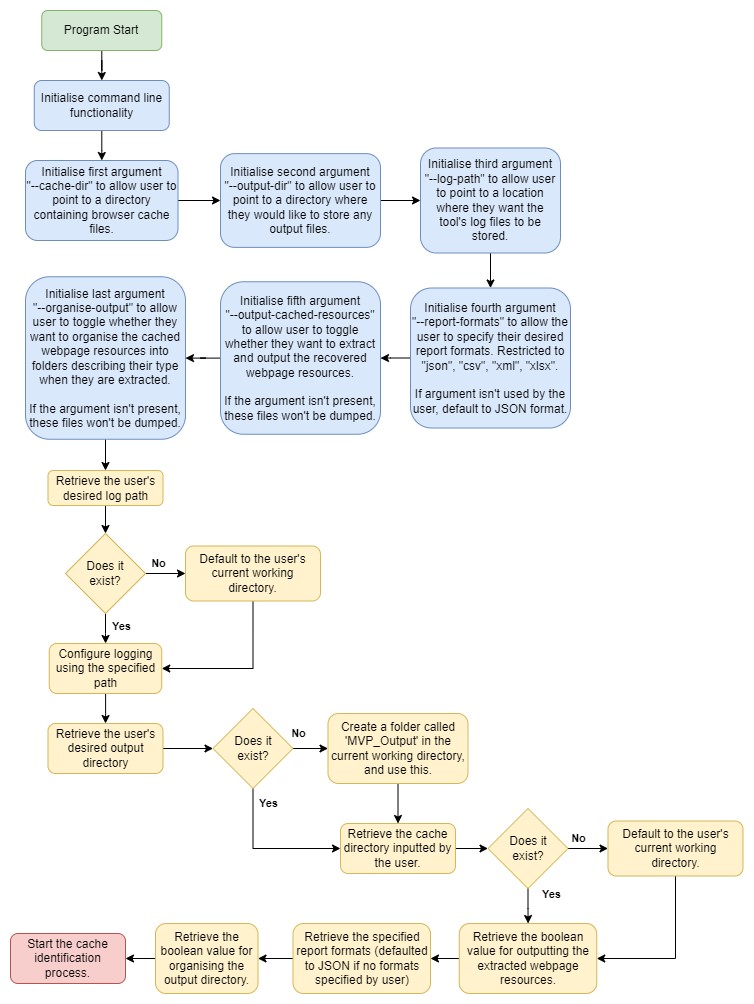
* 3.5.3.2 – (Must Have) Cache Recovery Mechanism)
* Go Back (Section 4.3 – Technical Artefact – MVP Cross-Browser Cache Parser



*20 The logic and processing flow behind decompressing the raw bytes of the requested resource.*

### 7.14 N CLI Integration Process

* 3.5.3.3 – (Should Have) CLI Integration)
* Go Back (Section 4.3 – Technical Artefact – MVP Cross-Browser Cache Parser



*21 Cache Parser CLI Integration Process*

### 7.15 O Reporting Mechanism Process

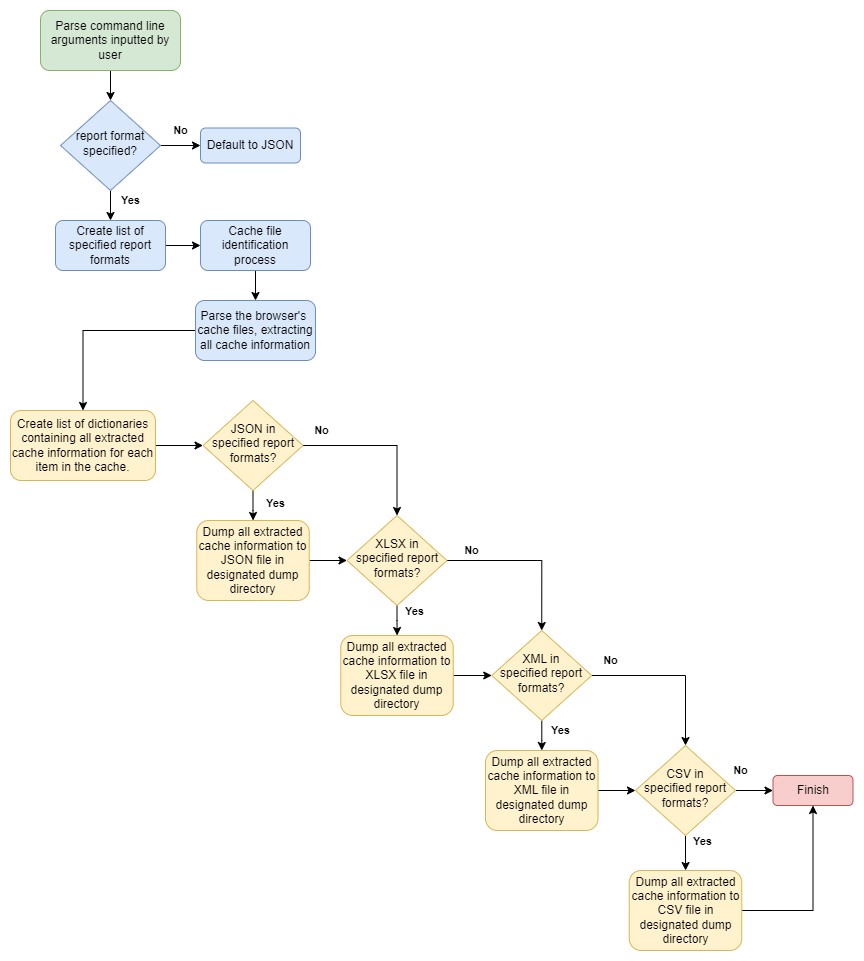
* 3.5.3.4 – (Should Have) Reporting Mechanism)
* Go Back (Section 4.3 – Technical Artefact – MVP Cross-Browser Cache Parser

*Figure*

*22*

*–*

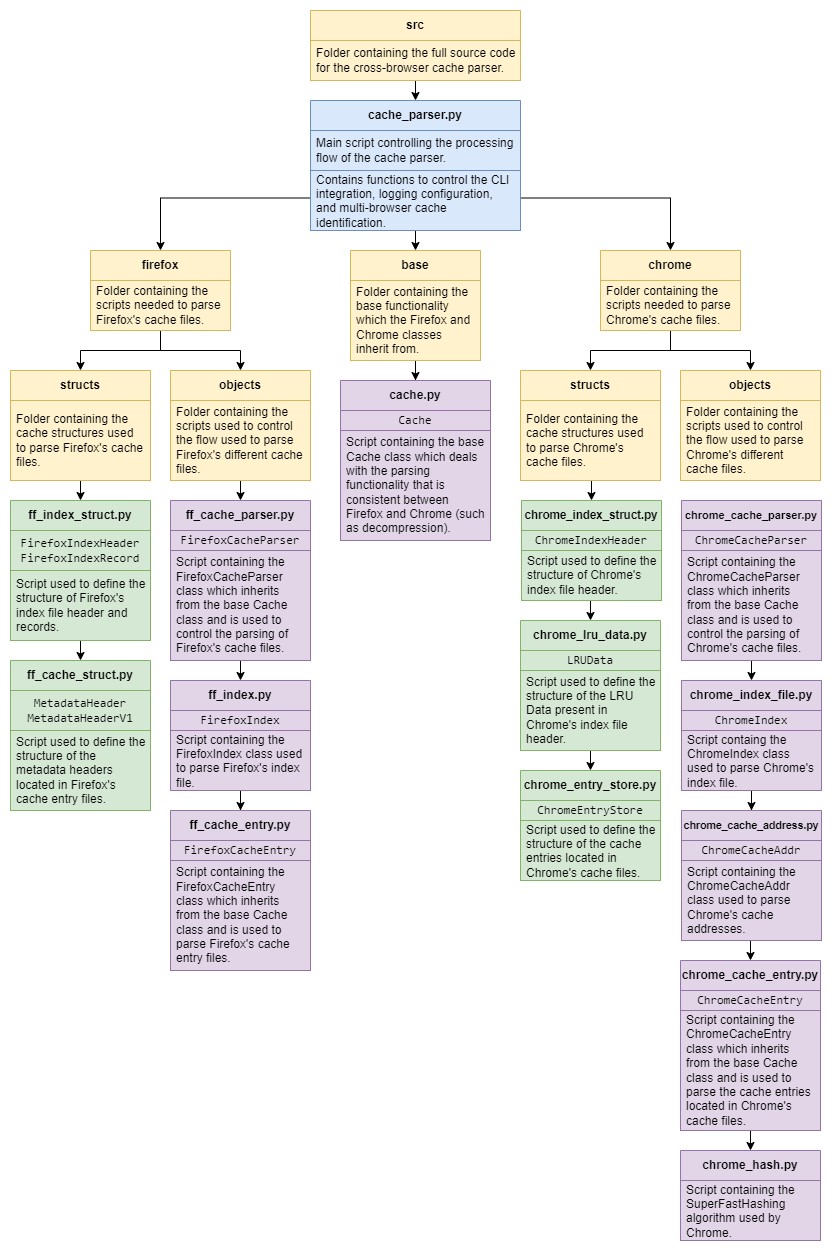
*Reporting Mechanism Process*



*Figure created by author using (diagrams.net, 2023).*

### 7.16 P High-Level Object-Orientated Cache Parser Code Structure

• 3.5.3.5 – (Could Have) Modular, Accessible, and Portable Code)



*Figure 23 - High-level overview of the cache parser's code structure, and the OO classes used.*

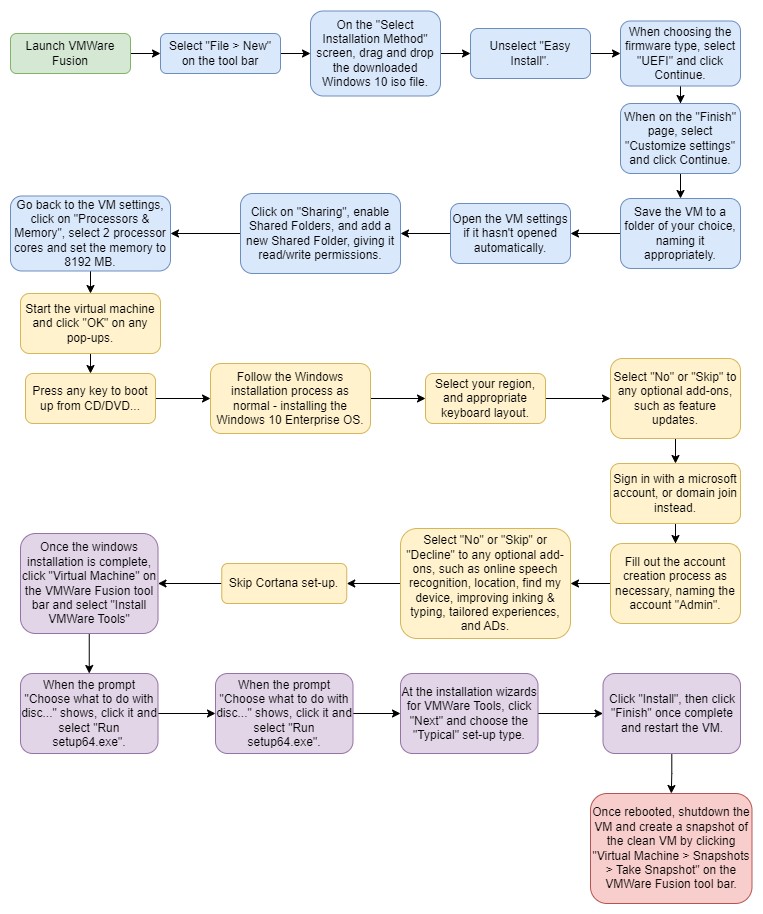
*Figure created by author using (diagrams.net, 2023).*

### 7.17 Q Environment Specifications and Set-up Process

* 3.5.4.1 – Environment Configuration Process)
* Go Back (Section 3.6.3 – Evaluation Process)

Table 27 defines the environments specifications.

Note – this process uses VMWare Fusion Version 13.0.1 to create the virtual environments.



*Figure 24 – The environment set-up process used to create all the project’s virtual environments.*

*Figure created by author using (diagrams.net, 2023).*

* Go Back (Section 3.5.4.1 – Environment Configuration Process)
* Go Back (Section 3.6.3 – Evaluation Process)

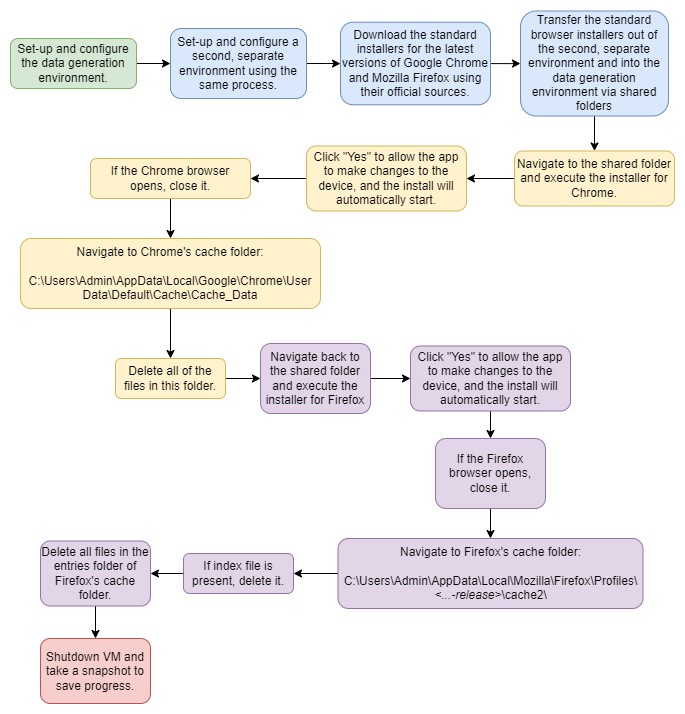
*Table 27 – Virtual Environment Specifications*

|  |  |
| --- | --- |
| **Spec** | **Value** |
| Number of Processors | 2 |
| Memory Size (RAM) | 8 GB |
| Hard Drive Size | 60 GB |
| OS | Windows 10 Enterprise Build 2009/20H2 |
| Network Mode | NAT |
| VMWare Tools Version | 12.1.5.20735119 |
| Shared Folders Enabled | Yes |

### 7.18 R Standard Browser Installation Process

* 3.5.4.1 – Environment Configuration Process)

Table 28 outlines the versions of Google Chrome and Mozilla Firefox used.



*Figure 25 – The full process used to install Google Chrome and Mozilla Firefox onto the data generation environment.*

*Figure created by author using (diagrams.net, 2023).*

* Go Back (Section 3.5.4.1 – Environment Configuration Process)

*Table 28 – Versions of Google Chrome and Mozilla Firefox used during data generation.*

|  |  |  |
| --- | --- | --- |
| **Application** | **Version** | **Source** |
| Google Chrome  (Normal Installer) | 111.0.5563.63  (Official Build) | <https://www.google.com/intl/en_uk/chrome/> |
| Mozilla Firefox  (Normal Installer) | 110.0.1 | <https://www.mozilla.org/en-GB/firefox/new/> |

### 7.19 S Webpages Selected for Data Generation

• 3.5.4.2 – Data Generation Process)

Table 29 contains the list of webpages that will be visited in the order they are presented.

*Table 29 – Table of websites visited to generate the data used for the evaluation of the cache parsers.*

|  |  |  |  |
| --- | --- | --- | --- |
| **Visit**  **Order** | **Website Name** | **URL** | **Watch Duration (excl. advert times)** |
| **1** | BBC | <https://www.bbc.co.uk/news> | N/A |
| **2** | The Guardian | <https://www.theguardian.com/uk> | N/A |
| **3** | YouTube | <https://www.youtube.com/> | N/A |
| **4** | TED – 7 new species of robot that jump, dance, and walk on water | [https://www.ted.com/talks/dennis\_hong\_ 7\_new\_species\_of\_robot\_that\_jump\_dan ce\_and\_walk\_on\_water/comments](https://www.ted.com/talks/dennis_hong_7_new_species_of_robot_that_jump_dance_and_walk_on_water/comments) | 1 minute |

### 7.20 T Cache Parser Commands

* 3.6.2.1 – Quantitative Evaluation Process)
* Go Back (Section 4.2 – Changes)

Table 30 provides a list of example commands used which will be used to execute the cache parsers.

Table 31 shows the cache folder paths which will be passed to the cache parsers.

*Table 30 – Example commands that will be used to execute the individual cache parsers.*

|  |  |
| --- | --- |
| **Cache Parser** | **Example Commands** |
| **ChromeCacheView** | chromecacheview.exe -folder “…” /sxml “…” /copycache “” “”  /CopyFilesFolder "…" /UseWebSiteDirStructure 0 |
| **MZCacheView** | mzcacheview.exe -folder “…” /sxml “…” /copycache “” “”  /CopyFilesFolder "…" /UseWebSiteDirStructure 0 |
| **MVP Cross-Browser**  **Cache Parser** | cache\_parser.exe –cache-dir “…” –output-dir “…” –log-path “…” -report\_formats xml –extract-cached-resources |

*Table 31 – Cache folder paths that will be used during the evaluation.*

|  |  |
| --- | --- |
| **Browser** | **Cache Folder Path** |
| **Chrome** | C:\Users\Admin\Desktop\Cache\_Artefacts\Google\Chrome\User  Data\Default\Cache\Cache\_Data |
| **Firefox** | C:\Users\Admin\Desktop\Cache\_Artefacts\Mozilla\Firefox\Profiles\  76ncd19y.default-release\cache2 |

### 7.21 U Quantitative Evaluation Metrics

• 3.6.2.1 – Quantitative Evaluation Process)

*Table 32 – Performance-based metrics collected from the cache parsers during the evaluation.*

|  |  |
| --- | --- |
| **Metric** | **Monitoring Tool / Method** |
| Total number of cached webpage resources identified | Count the number of entries present in the generated XML report. |
| Total number of cached webpage resources recovered | Count the number of recovered webpage resources that are outputted by the parser. |
| Percentage of process time used  *(% Processor Time)* | Windows Performance Monitor |
| Amount of memory consumed  *(Working Set – Private)* |
| Total IO Data Operations per second  *(IO Data Operations / sec)* |

### 7.22 V Qualitative Evaluation Questions

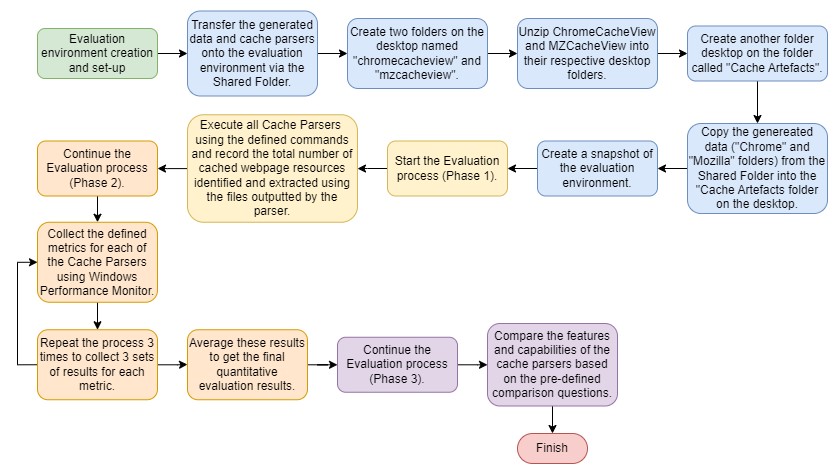
• 3.6.2.2 – Qualitative Evaluation Process)

*Table 33 – Comparative questions asked to qualitatively evaluate the capabilities of the cache parsers.*

|  |
| --- |
| **Comparison Questions** |
| What browsers are supported by the cache parser? |
| What report formats do the cache parsers support? |
| What information is extracted from the browser’s cache files? |

### 7.23 W – Full Evaluation Process

• 3.6.3 – Evaluation Process)



*Figure 26 - The evaluation process used to evaluate the performance of the cache parsers.*

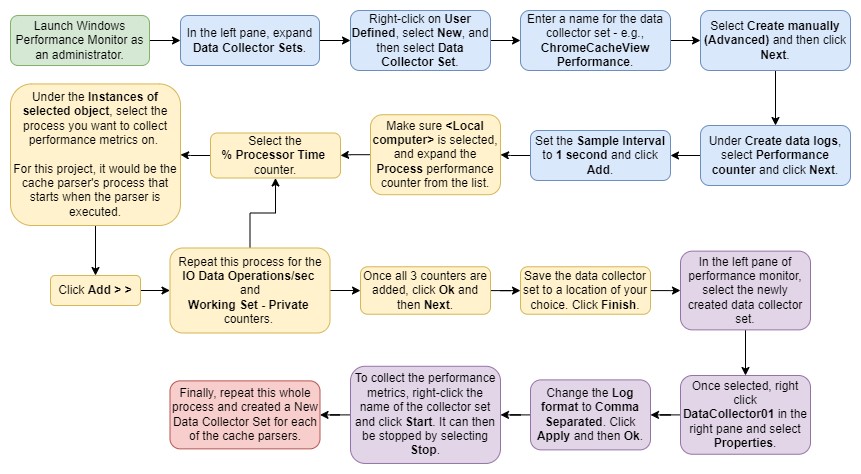
*Figure created by author using (diagrams.net, 2023).*

### 7.24 X Windows Performance Monitor

• 3.6.3 – Evaluation Process)

Figure 27 outlines the process used to collect performance metrics for each of the cache parsers.

This flowchart was based on the process outlined by (Tableau, n.d.).



*Figure 27 – The process used to collect performance metrics for each of the cache parsers during the evaluation.*

*Figure created by author using (diagrams.net, 2023).*

### 7.25 Y Evaluation Process Changes

• 4.2 – Changes)

Table 34 outlines the updated versions of the commands that were used for each of the metrics.

Table 35 shows the changed paths which were passed to the cache parsers during the evaluation.

*Table 34 – The actual commands that were used to execute the cache parsers during the evaluation.*

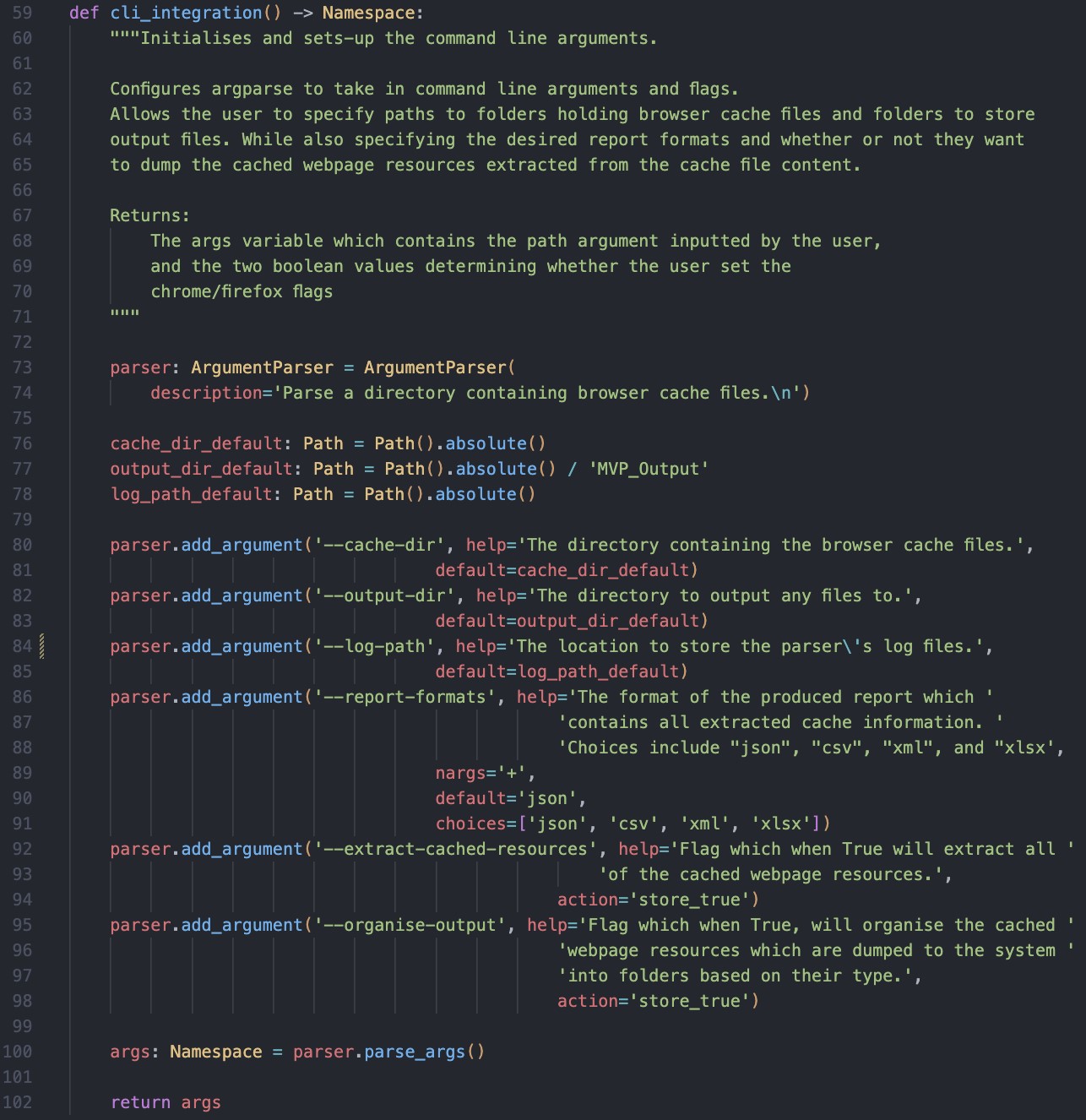
|  |  |
| --- | --- |
| **Cache Parser** | **Example Commands** |
| **ChromeCacheView** | chromecacheview.exe -folder “…” /copycache “” “” /CopyFilesFolder  "…" /UseWebSiteDirStructure 0 |
| chromecacheview.exe -folder “…” /sxml “…” |
| **MZCacheView** | mzcacheview.exe -folder “…” /copycache “” “” /CopyFilesFolder "…"  /UseWebSiteDirStructure 0 |
| mzcacheview.exe -folder “…” /sxml “…” |
| **MVP Cross-Browser**  **Cache Parser** | cache\_parser.exe –cache-dir “…” –output-dir “…” –log-path “…” – report-formats xml –extract-cached-resources |

*Table 35 – The actual cache folder paths that were used during the evaluation.*

|  |  |
| --- | --- |
| **Browser** | **Cache Folder Path** |
| **Chrome** | C:\Users\Admin\Desktop\Cache\_Artefacts\Google\Chrome\User  Data\Default\Cache\Cache\_Data |
| **Firefox** | C:\Users\Admin\Desktop\Cache\_Artefacts\Mozilla\Firefox\Profiles\  76ncd19y.default-release\cache2\entries |

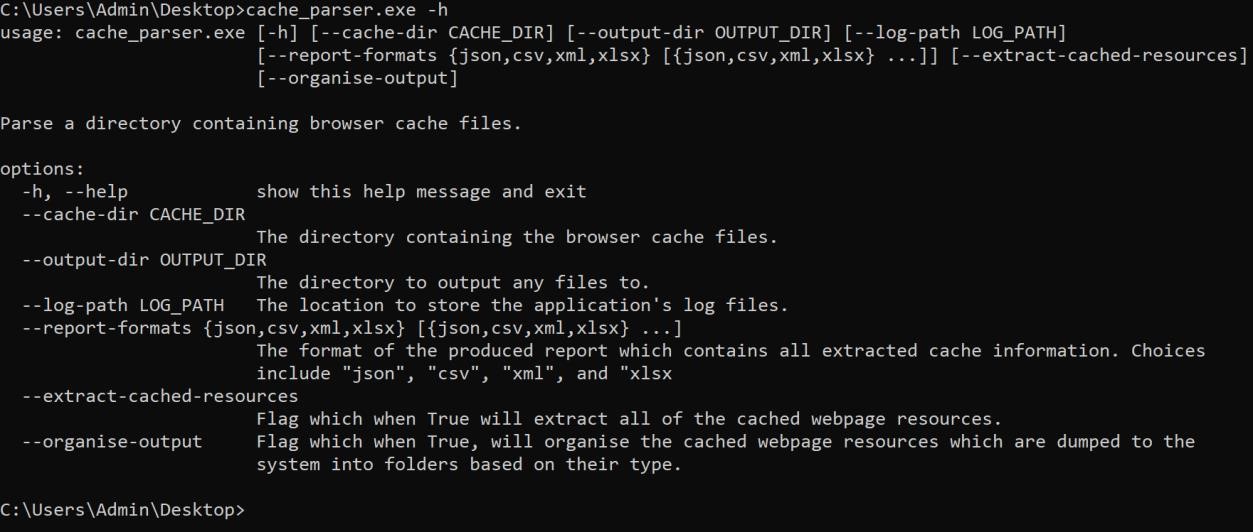
### 7.26 Appendix Z The Cache Parser’s Command-Line Integration

* Go Back (Section 4.3 – Technical Artefact – MVP Cross-Browser Cache Parser) Figure 28 shows the code used to implement the CLI Integration process into the cache parser. Figure 29 shows the cache parser’s help message.



*Figure 28 – The cache parser code used to implement the CLI Integration process.*

* Go Back (Section 4.3 – Technical Artefact – MVP Cross-Browser Cache Parser)



*Figure 29 – The cache parser’s help message displaying the arguments and flags it accepts.*

### 7.27 Appendix AA – The Cache Parser’s Cache Identification Functionality

• Go Back (Section 4.3 – Technical Artefact – MVP Cross-Browser Cache Parser)

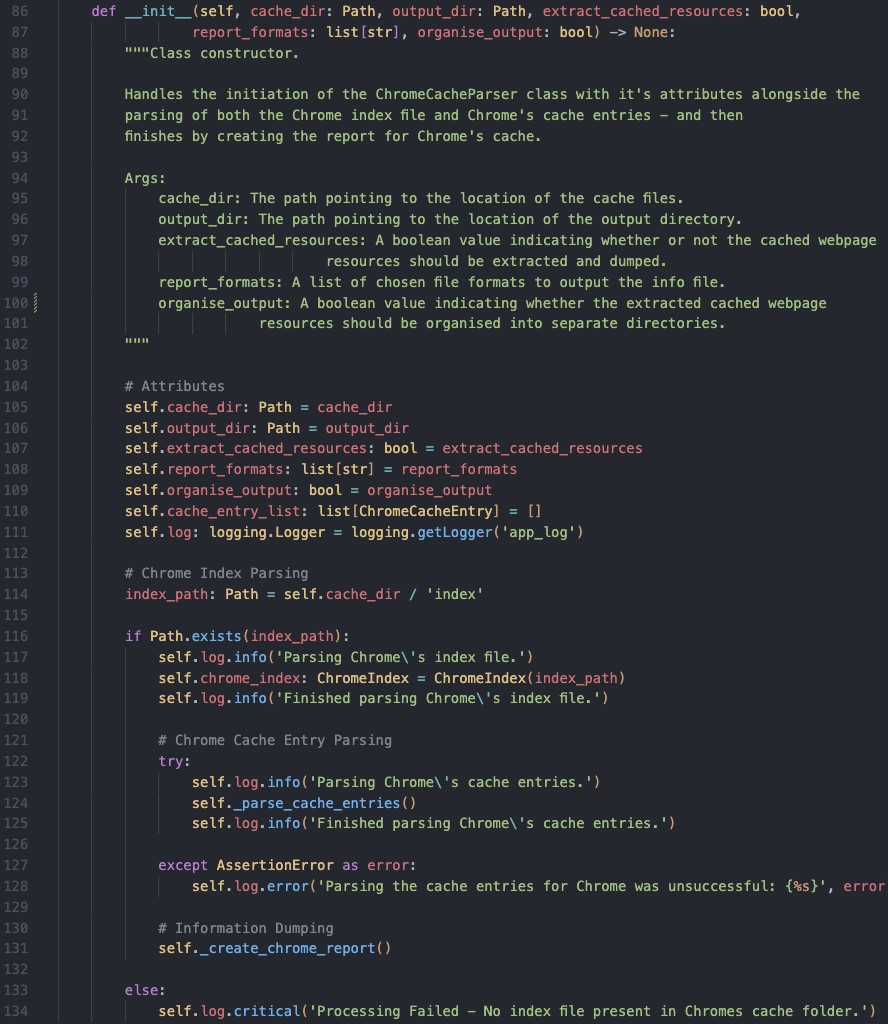


*Figure 30 – The cache parser code used to implement the multi-browser cache identification process.*

### 7.28 Appendix AB – The Cache Parser’s Chrome Parsing Mechanism

* Go Back (Section 4.3 – Technical Artefact – MVP Cross-Browser Cache Parser)

Figure 31 shows the code used to control the parsing of Chrome’s cache files. Figure 32 shows the code used to parse Chrome’s index file, and Figure 33 shows the code used to parse Chrome’s cache entries.



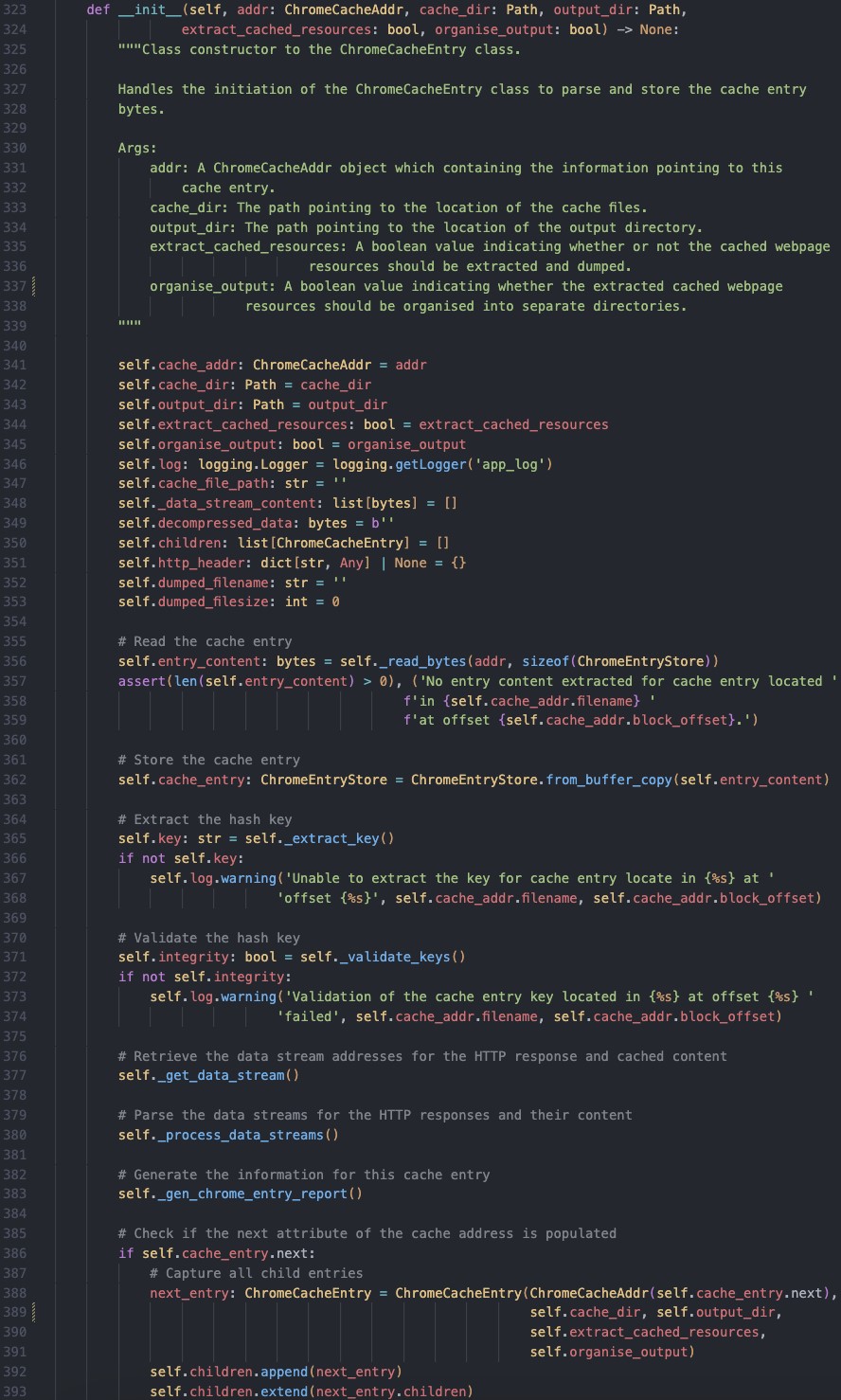
*Figure 31 – The code used to control the parsing of Chrome’s index files and cache entries.*

* Go Back (Section 4.3 – Technical Artefact – MVP Cross-Browser Cache Parser)



*Figure 32 - The code used to parse Chrome's index file.*

* Go Back (Section 4.3 – Technical Artefact – MVP Cross-Browser Cache Parser)

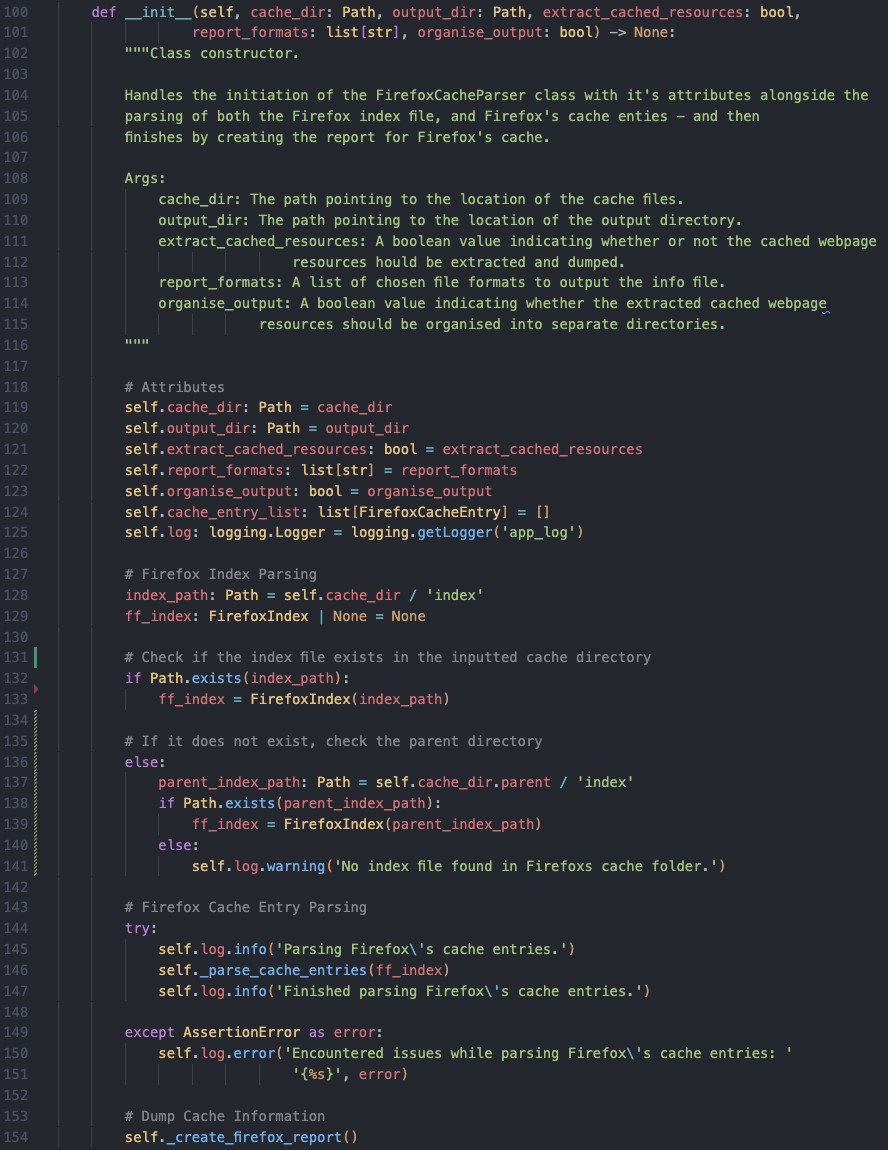


*Figure 33 - The code used to parse Chrome's cache entries.*

### 7.29 Appendix AC – The Cache Parser’s Firefox Parsing Mechanism

* Go Back (Section 4.3 – Technical Artefact – MVP Cross-Browser Cache Parser)

Figure 34 shows the code used to control the parsing of Firefox’s cache files. Figure 35 shows the code used to parse Firefox’s index file, and Figure 36 shows the code used to parse Firefox’s cache entry files.



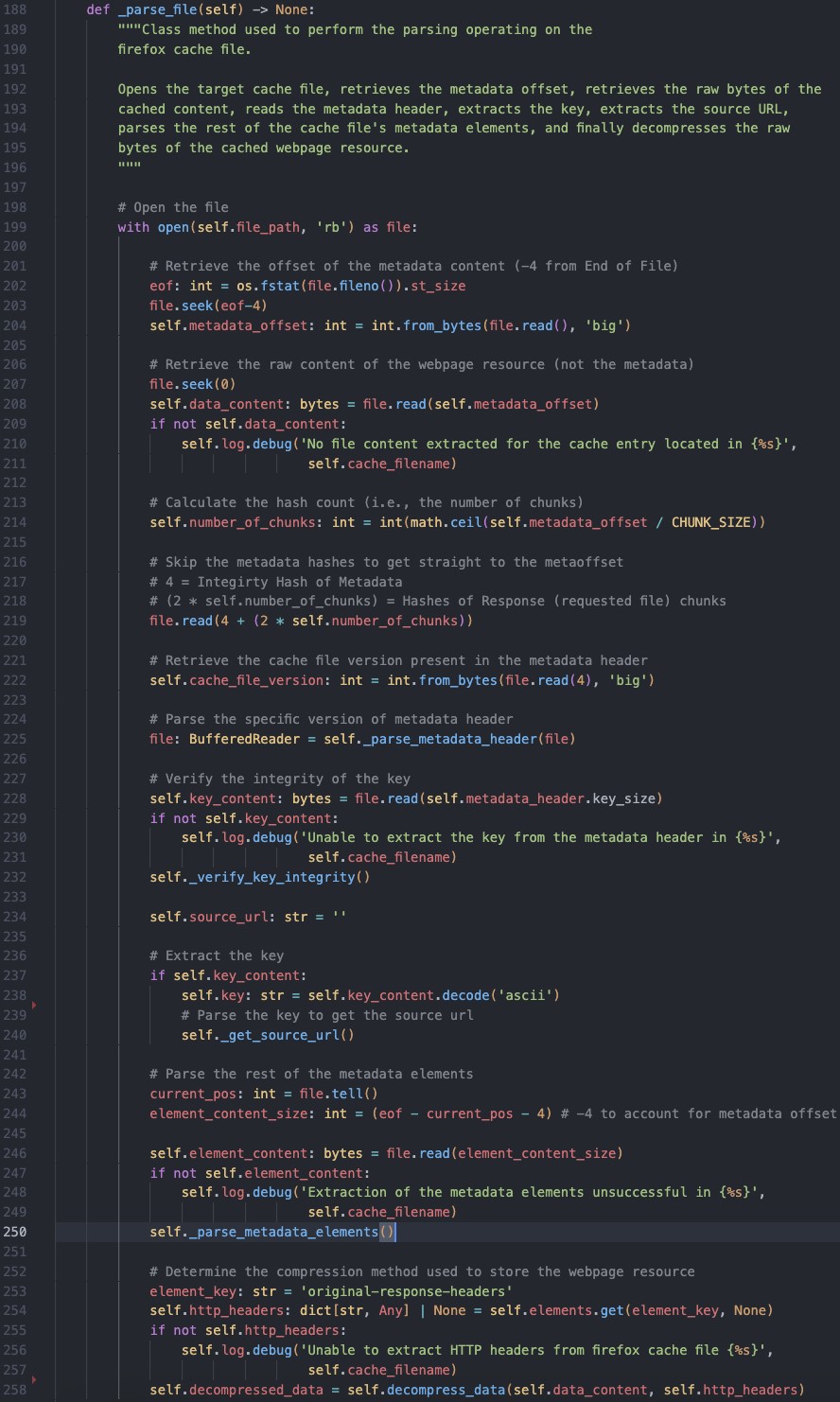
*34 The code used to control the parsing of Firefox’s index files and cache entry files.*

* Go Back (Section 4.3 – Technical Artefact – MVP Cross-Browser Cache Parser)



*Figure 35 – The code used to parse Firefox’s index file.*

* Go Back (Section 4.3 – Technical Artefact – MVP Cross-Browser Cache Parser)

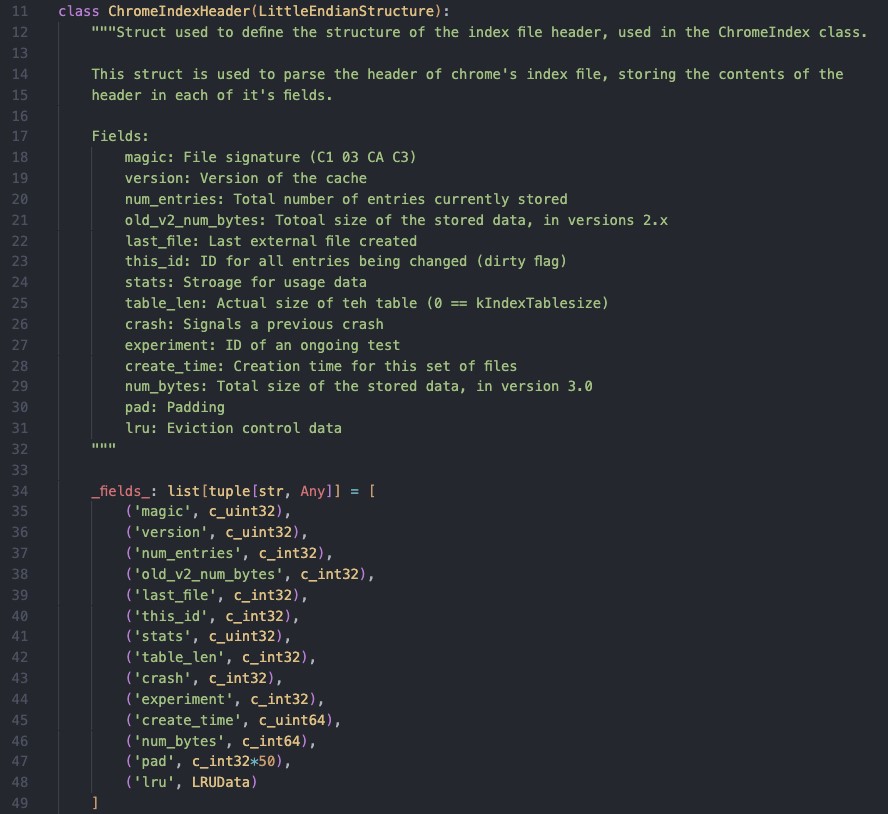


*36 The code used to parse Firefox’s cache entries.*

### 7.30 Appendix AD – Examples of The Cache Parser’s Cache Structures

• Go Back (Section 4.3 – Technical Artefact – MVP Cross-Browser Cache Parser)

Figure 37 shows an example of the cache structures defined in the code.



*Figure 37 - Example of the cache structures defined in the cache parser's code.*

### 7.31 Appendix AE – The Cache Parser’s Recovery Mechanism

* Go Back (Section 4.3 – Technical Artefact – MVP Cross-Browser Cache Parser)

Figure 38 shows the process used recover and recreate the cached webpage resource for Google Chrome.

Figure 39 shows the two functions used to output the recovered cached webpage resource.



*38 The code used to recover and recreate the cached webpage resources for Google Chrome.*

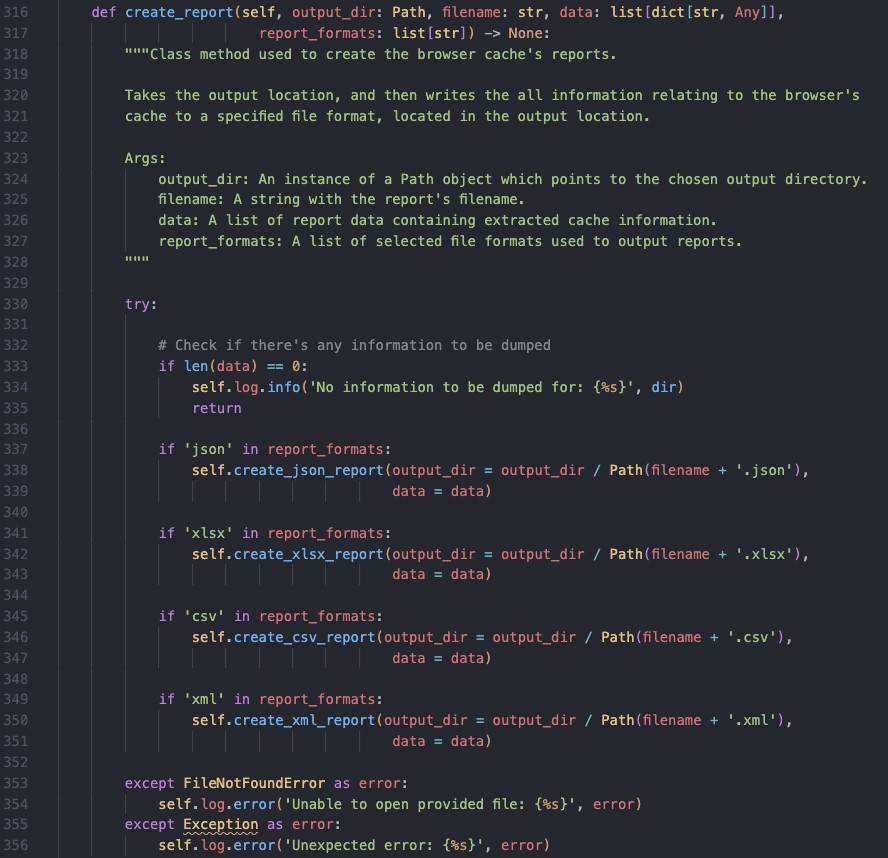
* Go Back (Section 4.3 – Technical Artefact – MVP Cross-Browser Cache Parser)



*Figure 39 – The code used to organise and output the recovered cached webpage resources.*

### 7.32 Appendix AF – The Cache Parser’s Reporting Mechanism

• Go Back (Section 4.3 – Technical Artefact – MVP Cross-Browser Cache Parser)



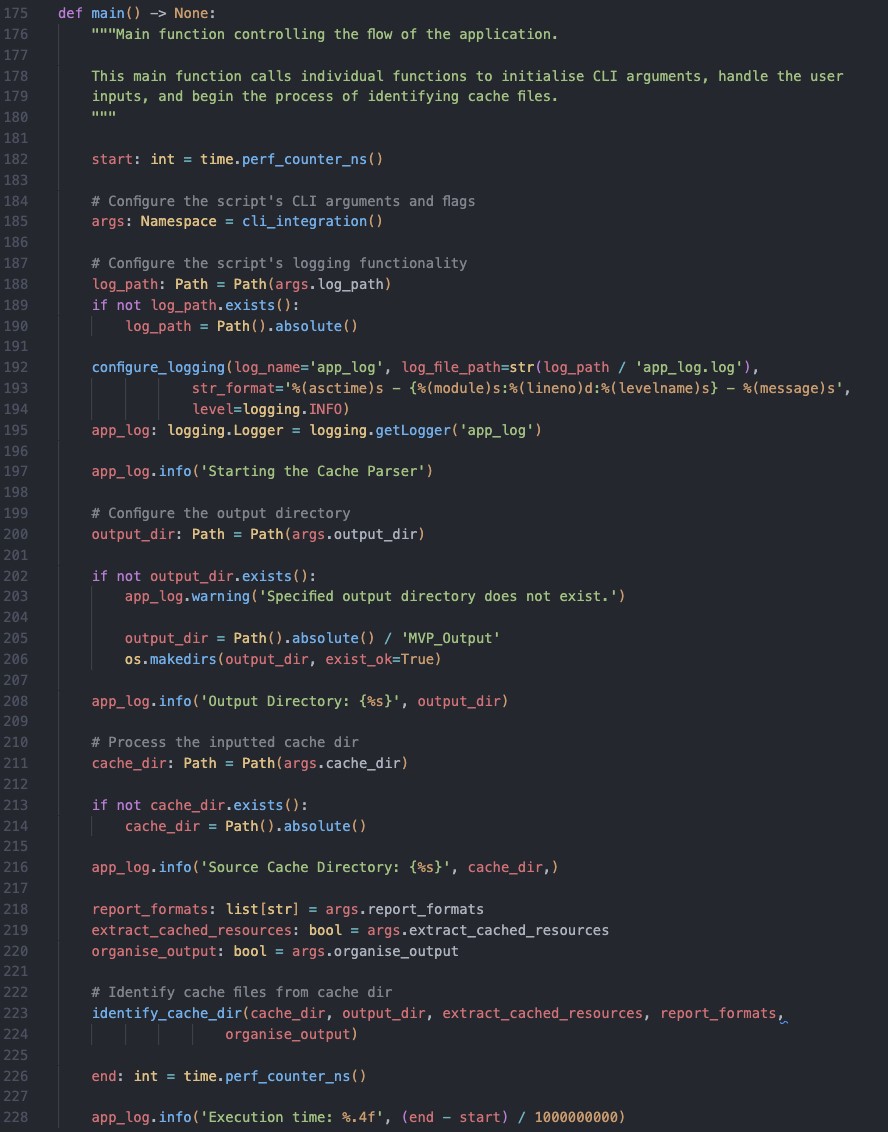
*Figure 40 – The code used to format the extracted cache information into a report.*

### 7.33 Appendix AG – The Cache Parser’s Error Handling and Logging

* Go Back (Section 4.3 – Technical Artefact – MVP Cross-Browser Cache Parser)

Figure 41 shows the initialisation of logging and examples of it being used to inform the status of the cache parser’s execution.

Figure 42 shows an example of what the logs looks like.



*Figure 41 – The code used to initialise the cache parser’s logging and output status updates to the log file.*

* Go Back (Section 4.3 – Technical Artefact – MVP Cross-Browser Cache Parser)

*Figure*

*42*

*-*

*An example of the cache parser's log file*

*.*



### 7.34 Appendix AH – Full Quantitative Evaluation Results

* Go Back (Section 4.4.1 – Quantitative Evaluation Results)

*Table 36 – Full results for the quantitative evaluation of the MVP Cross-Browser Cache Parser and ChromeCacheView against chrome cache files.*

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **MVP Cross-Browser**  **Cache Parser**  **(Chrome)** | **ChromeCacheView** |
| **Percentage of processor time used**  **(%)** | 1 | 77.80 | 32.34 |
| 2 | 97.44 | 32.58 |
| 3 | 92.26 | 20.58 |
| **Avg.** | **89.17** | **28.50** |
| **Amount memory consumed**  **(MB)** | 1 | 83.92 | 3.93 |
| 2 | 119.04 | 3.64 |
| 3 | 123.01 | 2.45 |
| **Avg.** | **108.66** | **3.34** |
| **IO Read/Write Operations per second** | 1 | 395.61 | 304.14 |
| 2 | N/A | 247.77 |
| 3 | N/A | 283.77 |
| **Avg.** | **395.61** | **278.56** |

* Go Back (Section 4.4.1 – Quantitative Evaluation Results)

*Table 37 – Full results for the quantitative evaluation of the MVP Cross-Browser Cache Parser and MZCacheView against firefox cache files.*

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **MVP Cross-Browser Cache Parser**  **(Firefox)** | **MZCacheView** |
| **Percentage of processor time used**  **(%)** | 1 | 89.22 | 61.70 |
| 2 | 90.70 | 44.37 |
| 3 | 89.91 | 27.33 |
| **Avg.** | **89.94** | **44.47** |
| **Amount memory consumed**  **(MB)** | 1 | 247.36 | 3.59 |
| 2 | 243.57 | 3.84 |
| 3 | 238.56 | 3.73 |
| **Avg.** | **243.16** | **3.72** |
| **IO Read/Write Operations per second** | 1 | 655.13 | 534.08 |
| 2 | 566.61 | 750.84 |
| 3 | 708.75 | 1022.56 |
| **Avg.** | **643.50** | **769.16** |