University of Plymouth

School of Engineering,

Computing, and Mathematics

COMP3000

Final Stage Computing Project

2020/2021

FORIN: PORTABLE DIGITAL FORENSIC INVESTIGATION

Jacob Alan Male

10579764

BSc (Hons) Computer and Information Security

I. Acknowledgements

Firstly, I would like to thank Nathan Clarke for his dedication in supporting the developer and the project. His questioning and reasoning helped guide the scope of the project. Secondly, I would like to thank Shirley Atkinson for supervising this year-long module. Her advice on project management proves vital in practice. Finally, I would like to thank both again for their support over the years for being my personal tutor.

Finally, I would like to thank all the lectures that have taught me over the years at Plymouth university, with a special thanks to all lectures in the final year, that provided great teaching standards, lecture interactions, and feedback despite the Covid 19 pandemic.

II. Abstract

This report contains the application of digital forensics in a portable environment. Using python3 and open-source tools to develop a wizard application designed for public use on Linux distributions. The process of traditional digital forensics can be described as overwhelming and complicated. Users generally have no clue on how they can recover their own data, while investigators have many tools they must recognise and use during an investigation. This process can be simplified for users by combining all the tools that digital investigators use to process and analyse the data. using a low-power ARM computer to represent a portable device, analysis of these tools was also conducted. This allows for an understanding on whether these tools and this process could be applied to traditional investigation or the end user.

The report will first contain the relevant background information, describing all relevant context related to the project. This describes the process while also describing aspects required for understanding later in the report. From this, the objectives from the report are described.

This is continued by a relevant thorough look and legal, social, ethical, and professional issues. Describing some examples of relevant laws and their consequences while also ensuring correct legal steps are taken to not fringe any licences of the application. Continuing, an ethical analysis was performed to ensure the application will leave a positive impact.

Next, the management of the project and what technologies were to be used are described. Descriptions of the management approach and how it should be discussed are presented, while describing what version control methods were used. A description of the technologies used and why they are used is also included.

Continuing, the requirements of the application are then present based on what is believed to be the best forensic tasks to include. The development of the project followed the requirements and included descriptions of each different stage of development (Early/Mid/Late-Stage Development)

Finally, there is a discussion on evaluating the project. The evaluation is based on usability testing and functionality testing with critical analysis compared to the results. Testing of tools that were to be used was also included to ensure that this type of project would be viable on a portable device.

The closure of the project will include a project end-closure report and a post-mortem to critically analyse both the success and failures of the project. Highlight what when wrong and what was the solution when compared to requirements, testing, and the overall viability.

III. Table of Contents

I. Acknowledgements ... 1
II. Abstract ... 2
III. Table of Contents ... 3
IV. Table of Figures ... 4
V. Table of Tables ... 4
1. Introduction ... 6
2. Background ... 6
2.1. Problem & Solution ... 7
2.2. Objectives & Deliverables ... 8
3. Legal, Social, Ethical, Professional ... 10
3.1. Legal ... 10
3.2. Social ... 11
3.3. Ethical ... 11
3.4. Professional ... 12
4. Project Management ... 12
4.1. Office 365 Planner (Management) ... 13
4.2. GitHub & Google Drive (Version Control & Backups) ... 13
5. Technologies ... 14
5.1. Python3 ... 14
5.2. Kali Linux ... 14
5.3. Raspberry Pi 4 ... 15
6. Design & Architecture ... 15
7. Requirements ... 18
8. Development ... 19
8.1. Sprint 0 (Project initiation & MVP) ... 20
8.2. Sprint 1-5 (Early-Stage Development) ... 20
8.3. Sprint 6-10 (Mid-Stage Development) ... 22
8.4. Sprint 11-15 (Late-Stage Development) ... 24
9. Evaluation & Testing ... 26
10. Project End-Report ... 30
10.1. Requirements & Objectives ... 30
10.2. Application Viability ... 32
11. Project Post-Mortem ... 32
11.1 Project Management & Approach ... 32
11.2 Application Code & Design ... 33

11.3 Developer Performance 33
11.4 Future Developments 33
12. Conclusions 34
VI. References 35
VII. Appendices 37
Appendix 1: Installation & Usage Guide 37
Appendix 2: Planner & Sprints 39
Appendix 3: User Testing Documentation 41
Appendix 4: Usage UMLs 44
Appendix 5: User Testing Results 52

IV. Table of Figures

Figure 1: Hashcat CLI usage example. 7
Figure 2: Agile Software Development 12
Figure 3: Minimum Viable Produce 13
Figure 4: Traditional deployment example 16
Figure 5: Forin app deployment example 16
Figure 6: Initial Usage Diagram 17
Figure 7: Forin app SSH deployment example 18
Figure 8: Raspberry Pi4 Boot Load Error Screen 21
Figure 9: Pi4 SD card performance 22
Figure 10: Pi4 SSD performance 22
Figure 11: Early Settings Config file 23
Figure 12: TerminalMenuScript.py, generate_promt_menu function. 23
Figure 13: test_fsisetting_class.py, test_setting_class function 25
Figure 14: FSI, File System navigation implementation. 26
Figure 15: Forin Application UML Diagram 27
Figure 16: Forin Questionnaire Question 1 Results 28
Figure 17: Forin Questionnaire Question 2 Results 28
Figure 18: Forin Questionnaire Question 3 Results 28
Figure 19: Forin Questionnaire Question 4 Results 28

V. Table of Tables

Table 1: Forin Application Objectives 8

Table 2: Forin Application Deliverables 9

Table 3: Forin application Client Objectives 9

Table 4: Raspberry Pi 4 B Specifications (Raspberry Pi Foundation, 2021) 15

Table 5: Forin Functional Requirements 18

Table 6: Forin Non-Functional Requirements 19

Table 7: Forin Application Minimum Viable Product 20

Table 8: Drive Acquisition Testing Results 29

Table 9: Data Craving Test Results 29

Table 10: Hash Cracking Test Results 29

Word Count: 10956

Git-Hub Repository: https://github.com/Jiskey/Raspberry-Pi4-Forensics

1. Introduction

At the scene of a crime, a forensic investigation allows understanding of both what happened and who is responsible, applying scientific investigatory techniques to crimes and attacks. (Fruhlinger, 2019) Analysing evidence in traditional crime forensics allows law enforcement to identify a possible suspect. Digital forensics is no different to this idea instead, it looks at digital evidence to construct a timeline of events. Cyber-attacks on users and companies can lead to fraud, lost data, etc, amounting to huge consequences that require a quick and urgent response. Digital devices such as mobile phones and servers, all contain data and capturing this data from a suspect device allows investigators to analyse it, checking metadata, in-file contents, network traffic and more, with the intent to identify perpetrators.

Traditionally, the digital evidence is collected from the site and sent back to a lab for analysis using a set of specialty tools. The tools themselves will require a large amount of processing power and are executed on computers with top of the range GPU & CPU performance capabilities. When one suffers an attack, a problem arises because most will not have these types of machines to perform forensic tasks due to cost and accessibility, most users not needing a GPU, with only around 40% of computers containing discrete GPUs as of 2017, not accounting for the range of price these devices can cost which can be upwards of thousands. (Jonpeddie.com. 2017). The concept of performing portable digital forensics is nothing new, physical portable 'Forensic Imagers' do exist but again are expensive, also requiring experience and only performing a single task. Looking at concepts such as Kali Linux Net Hunter (A Mobile Phone Penetration Testing Kit), And small ARM devices such as the Raspberry Pi4, or Librem 5 Mobile Phone getting more powerful and user friendly, shows the possibility of performing digital forensic tasks quickly and easily from already circulating devices.

This project studies the performance, abilities, and operation of free, open-source digital forensic tools on a portable device. The main objective of the project will be to create a wrapper that will combine the multiple tools available to perform different digital forensic tasks allowing for quick and easy use for these tools. Furthermore, looking at the capabilities of these portable devices to understand how they can be used to advance companies and users in their digital investigations. Performance testing will also be conducted to evaluate the overall viability of these portable devices with use of an ARM single board computer.

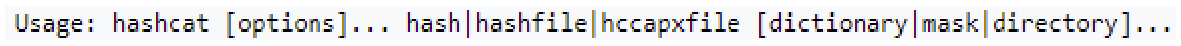
2. Background

Digital data, like forensic evidence at the scene of a crime, is highly volatile and if not handled correctly can be contaminated making it inadmissible in court. When collecting data, the computers state, the data's integrity, and the type of attack all determine different difficulties for investigators. Data corruption means data may not be recovered and reconstructed, capturing only some aspects of the file, and if the computer is off, any volatile data such as memory stored in the RAM is lost. Furthermore, this does not just happen to businesses and organisations, everyday users are subject to attacks online with an estimated number of attacks being over 80,000 daily in 2018 (Purplesec.us, 2019), many people believe it is the end for their data and this is simply not always the case. The expensive option is a user may opt to call a professional data recovery company to retrieve the data, or they may accept the damage completely, either is not ideal for users. Though some tools used in digital forensics are proprietary and require payment, there are many tools that are free, open-source, easy-to-use, and available to download on different operating systems, allowing for the possibility for individuals to attempt to retrieve their data themselves.

Different actions in digital forensics require different and sometimes multiple tools to perform such actions. Because of this, an investigator is required to know and understand how many tools operate to use them to the best of their ability. Tools can include,

- 'Dc3dd' or 'Dcfldd' for drive acquisition.
- 'Hashcat' for password cracking.
- 'Sleuth kit (TSK)' for file system inspection & extraction.
- And many more...

These tools are all open-source and can be downloaded for Linux, but some are available for windows and most follow the similar usage by using the command line interface (CLI) to execute and command.



```
Usage: hashcat [options]... hash|hashfile|hccapxfile [dictionary|mask|directory]...
```

Figure 1: *hashcat CLI usage example. (Steube, J. and Gristina, G., n.d.)*

Using this format is quick and easy if you know how to use CLI tools, with new tools requiring a simple help page lookup to learn. However, all these tools follow this format meaning though easy to use, can be hard to remember and can lead to organisation issues if the volume of tools used or the volume of data used is large. Companies Employ strict workflows to combat this, such as a chain of custody that tracks who has had access to the data during the investigation. This process can be simplified, cheapened, and quickened to benefit both organisations and the individual user.

2.1. Problem & Solution

The primary physical problems with digital forensics are time, expense, and equipment. As digital media devices get larger in capacity, and more services are moving to the cloud, merely acquiring the data can be time-consuming and the hardware required is quite expensive. Using devices that are already in circulation like mobile phones, could decrease the time it takes to acquire multiple suspect drives for example, by capturing more drives at one time. Because of this, different tools will perform better based on different aspects of the machine, meaning drive acquisition depends on the ports used (read and write speeds) whereas password cracking requires a hefty GPU allowing for fast math calculations. Understanding this is important to prioritizing which tools are best suited for these portable devices and the target audiences.

The solution is to create a wrapper (Forin) with the python programming language to perform digital forensic tasks using the relevant tools. Creating the project with design specifications in mind to ensure that the application is accessible as can be to everyone. From a CLI (Command Line Interface), users will be able to easily conduct digital forensic tasks using multiple tools from a single wizard type software. It will be designed to be used by both investigators and common IT users alike to simply retrieve their data or perform a quick investigation. The application will require no internet connection and be fully contained on the device in which it is installed due to potential security risks it involves when using the tools. This will allow users to easily deploy the application from a range of devices such as the Pi4, a USB stick, or even remote access via SSH.

2.2. Objectives & Deliverables

The aim of the project is to create a CLI application that can perform a range of forensic tasks. In digital investigation there are many tools and aspects, to keep the project with scope, an 'offline' approach is taken. "Live forensics gathers data from running systems, providing additional contextual information that is not available in a disk-only forensic analysis." (Adelstein, F., 2006), staying primarily with traditional forensics. The application should guide the user on what they want to do and how they can do it by using a wizard format to execute the tool for them. Using simple CLI inputs and asking the user a series of questions to customise their execution without having to remember the tool's CLI commands themselves. The tables below show requirements of the project application, looking at accessibility, requirements, and usability.

FORIN APPLICATION OBJECTIVES

ACCESSIBILITY	• Free and open source. not requiring any closed-source or limiting software/hardware. • Linux based application programmed in Python3. • Application should function on multiple deployable platforms such as a virtual machine or a portable ARM device.
RELIABILITY	○ The application should have little to no errors. ○ The application should have correct error handling. ○ The app should not require internet access and be isolated on the host. ○ The application should execute the correct tool commands.
DATA	• The app should not store any personal information and only store information provided by the user. • The app should store evidence correctly and without alteration. • The application should record actions performed by it.
USABILITY	○ Users should be able to execute multiple different tools to perform multiple different tasks. Including, but not limited to. ○ Drive Acquisition ○ File System Inspection ○ Data Carving ○ PDF file analysis

	○ Users should be able to retrieve data from a digital drive. ○ Users should be able to retrieve and/or analyse deleted data. ○ Users should be able to attempt to retrieve and/or analyse corrupted data. ○ Users should be able to analyse PDF files. ○ Users should be able to specify different options within the application.
EFFICIENCY	• The application should have low performance requirements. • The application should be efficient, following good programming practice.
SECURITY	• The application must not execute any super-user commands without specific authorisation from the user. • The application must not edit any data or files outside any program or user specifications. • The application should not exploit any host machine vulnerabilities. • The application should only use trusted and safe tools to perform its operation.

Table 1: *Forin Application Objectives*

FORIN APPLICATION DELIVERABLES
• An interactive application designed for use within digital investigation.
• Command Line Interface (CLI) for multiple deployment options.
• Execute CLI commands to perform forensic tasks with use of open-source tools.
• Application is easy to use for both novice and advanced users.
• Contain variable and customisable options for tools and application.
• Uses Debian 10.x (Linux) as base OS. (Development on Kali Linux.)
• Uses Python3 as the programming language and operation.
• Uses Python package imports to function.
• No additional hardware required (Software Only)

Table 2: *Forin Application Deliverables*

The application must be created with the intent to use only free and open-source tools while also being easy to use. The benefit of this is that it can help individuals who may not be so literate in the field, while also not being tedious for IT experts.

FORIN APPLICATION CLIENT OBJECTIVES
• Be able to conduct digital investigation techniques with the use of the application, generating commands for the selected tool.
• Be able view and log actions that the application has executed.
• Be able to collect a forensic image of the drive connected to the device.
• Be able to collect and view deleted files off the collected forensic image.

- Be able to attempt to collect and view corrupted data off the collected forensic image.
- Be able to use the application to assist in usage of trusted digital forensic tools.

Table 3: *Forin application Client Objectives*

3. Legal, Social, Ethical, Professional

The Application will be open-source and free for people to use however they like. Variables such as the license used, and the packages installed may include certain restrictions on how they can be used or if they can be used at all. Alongside the nature of the program that includes tools that can be used for both felonious and innocent actions, releasing this application publicly could pose some serious problems if not grasped correctly.

3.1. Legal

The legal concerns for this project are mainly to do with licencing however, users' usage of the application can raise some concerns and should be addressed. When collecting evidence from the scene, the relevant authorities will require a search warrant, in which paragraph 7.7 of the Police and Criminal Evidence Act 1984 (PACE) Code B states, "The Criminal Justice and Police Act 2001, Part 2 gives officers limited powers to seize property from premises or persons so they can sift or examine it elsewhere" (Police and Criminal Evidence Act 1984 (PACE) Code B 2013, Section 7.7) This would be applied if officers were collecting evidence where the suspect property would be a USB stick containing digital evidence, sifted at a IT forensic lab. The app is a wrapper and allows users to 'easily' accomplish these tasks and instead uses other tools to perform the tasks. Without a search warrant, the app should not be used on anything, but the users own personal data, unless given correct consent by the relative authority. The Computer Misuse Act 1990 states if the user "causes a computer to perform any function with intent to secure access to any program or data held in any computer" (Computer Misuse Act 1990, c.18, S.1 (1)), then the perpetrator could be found guilty of misusing the data.

As the project is designed to collect and examine different types of data on any device, it will use a bundle of different tools already available to accomplish its goals. Licensing this project correctly will ensure that the author does not break any copyright, or distribution laws. Meaning the project will be realised under the MIT license (Opensource.org. n.d.). a license that allows users to freely use the application, using it for whatever they like but importantly, also protects the author from liability claims.

Finally, it is important to follow the licenses of the tools and technologies used. Python will be the programming language used for development. The Python Software Foundation licence agreement for python 3.9.5 (latest at time of writing) section 2 explains that python can be used and distributed completely royalty free (Rossum, G., 2021). Furthermore, the tools used will also have a licence connected to them such as the Apache 2.0 License. Under section 4(a) of this licence, it states that all recipients of this project must receive a copy of this license (Apache.org. 2004.). Following these rules is imperative when it comes to distributing this project to the public under the MIT license.

3.2. Social

When developing software, it is important to not exclude any social groups from participation or target such groups in any way through the application. An obligation must be made to ensure that any acts against race, community, gender, age, and disability are not included from the start of development and followed indefinitely, unintentionally, or not, through comments or design. Because of this, the program will tackle these problems by the following:

- Gender/Race Neutral – There should be no mention of gender or race at all within the application.
- CLI – Uses a simple CLI interface for operation, not using any equipment that could be troublesome for users with disabilities.
- User Help – Application will contain necessary information for less knowledgeable users, not only focusing on professionals.
- No Unique/Discriminating Hardware – All hardware required by the software is either free or at least low investment (Less Than £100) and is accessible to everyone.

As the software will be free and open source for everyone to download, following these social rules are of top importance. Aside from this, the nature of the program could pose a social issue. Deploying such a project on a portable device could allow users to easily collect data from drives in places where security may not be up to par. To combat this, the MIT License will be used and ensure the author is NOT liable for damage caused by this program (intentional or otherwise) on the user themselves or others and is realised "as is".

3.3. Ethical

There is a huge ethical issue when it comes to creating a wrapper for tools used in the digital forensic field. Firstly, these tools as previously mentioned, can be used with malicious intent, a user could theoretically use the app to acquire information with no prior knowledge. Known in the industry as 'script kiddies', Patrick Putman explains that "Often a script kiddie will use these programs without even knowing how they work or what they do." (Putman, P., n.d.). Though, it is believed that the benefits of this tool out-way the negatives. Firstly, an argument could be made that these tools already exist in the public domain and do not require much time to learn. The project attempts to speed up this process by both deploying it in a portable fashion and encouraging users with its simplicity. Secondly, a wrapper like this could be used for educational, professional, or personal use due to the licence, making it available to everyone, not just malicious users.

The project will also include a user testing element after completion. To do this correctly, the relevant procedures and policies must be followed. As the covid-19 pandemic is affecting studies across the world. University of Plymouth policy was updated to assist students with this and now includes family and friends as part of the participation list, as of the Faculty Ethical Approval Guidelines December 2020. As such, two documents were given to testers that stated their right to withdraw and the contents, alongside an anonymous questionnaire (Appendix 3).

3.4. Professional

Professional standards should be followed throughout the development of the project. Coding standards such as code-naming conventions should be relevant and appropriate, following a similar pattern throughout the application for easy code readability and maintainability. Furthermore, Adequate testing procedures should be followed, including all aspects from application to usability testing. Correct version should also be followed, using trusted and professional web-services, maintaining good version control assists in reading code changes and additionally makes maintainability easier.

As the application will be open source, a professional and appropriate license should be included on release. It is important to understand the appropriate environments in which it could be deployed. Though anyone can use it, it will not be a completely robust system and may not be fully secure and may not be appropriate to be released in a legal or commercial setting for example. Arguably this is because of the educational setting of the project and may include possible security flaws. Nevertheless, the open-source nature ultimately decides how the application is used.

4. Project Management

The project will be following a SCRUM approach to development. This is because after a single tool is completed, its implementation can be tested, and a deployable version of the application can be made. Through development there will be multiple stages of design, development, and testing as multiple different types of tools will be used. The reason for applying an agile approach is because the data from one tool to the other will rarely be compatible with another. Meaning each tool will have to be designed, developed, tested, and maintained individually. There was also use of cloud-based services for sprint planning, alongside bi-weekly meetings with the project supervisor across VOIP and posts on message boards, correctly applying the appropriate procedures due to the Covid 19 pandemic.

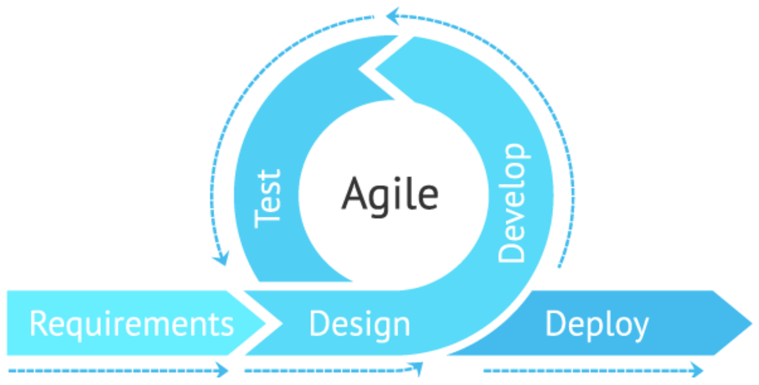


Figure 2: *Agile software Development* (Hazevytch, L. and Vilchynska, H., 2020)

It can also guarantee a release of a minimum viable product (MVP). A minimum viable product is the idea of developing the software in which a project could be deployed at any time in case of reasons preventing further development. Furthermore, implementing the most important aspects of the application first means that the primary intended function is created first before any extras or additions are added. The minimum viable product for this project will contain the most crucial aspects of traditional digital forensics and was decided to follow the agile development process of the project. Each step will be fully integrated and tested in no specific order to achieve a consistent and reliable MVP throughout development and must be integrated with a functional CLI wrapper that correctly functions and executes commands without hindrance.

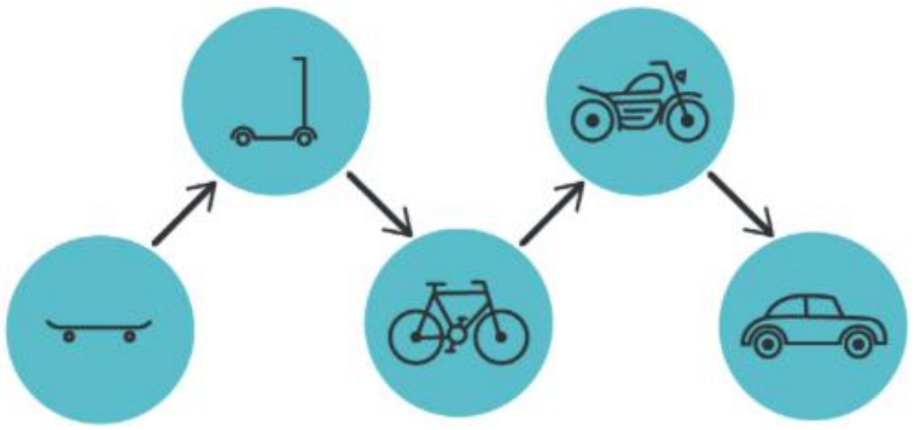


Figure 3: *Minimum Viable Product (Business Model Toolbox. n.d.)*

4.1. Office 365 Planner (Management)

To manage the project, Office 365 Planner will be used. This interactive planner allows you to create buckets or sprints and assign each of them several tasks to be completed by a certain deadline. The addition of a backlog too allows for ideas to be stored and potentially used later. However, if something were to delay a sprint or two from being completed, the planner can help reorganise oneself and following and maintaining the plan through development allows for easy management and organisation of tasks. These tasks can be anything from creating a document, to adding a piece of core functionality to the application.

4.2. GitHub & Google Drive (Version Control & Backups)

GitHub and google drive will be used to handle backups, version control, and data holding. Version control is the practice of tracking, managing, and recording changes to software code (Bit Bucket. n.d.). GitHub will be used in the CLI to clone, commit, push, and pull the project, allowing for quick deployment on multiple different devices, crucial for testing across multiple platforms and versions. Each commit to GitHub should have an appropriate name describing what the commit is and should be used whenever a code-based task is completed, being consistent throughout the development stages.

However, a drawback of GitHub is its inability to commit large files, such as drive images. This is perfectly fine as large data storage is not what GitHub is supposed to do. Turning to an online cloud-based service to store files and data that may be needed. Google Drive is one of these options and allows the user to use 15GB of free space, allowing the upload of large drive image files or test results, whilst being able to quickly retrieve them while operating in different environments.

5. Technologies

The application will have to be simple and easy to use, primarily due to over complications that can lead to discouragement of use. Use of the Linux Terminal to create a CLI application allows for ease of use and flexibility in portability. It could be argued that a GUI would be preferred but is argued that deployment could be limited due to the GUI, or the scope of the project is never reached due to extra GUI implementation. Developed for Linux, using Python3, a wrapper program to integrate usage digital forensic tools will be developed.

5.1. Python3

Python is a scripting language well known in the IT industry. Developed originally by Guido van Rossum and released in 1991, It later became the scripting language of choice by many around the world. The reason for this is Python's high-level flexibility, supporting a wide range of standard libraries including string processing, Software Engineering (unit testing, logging.), and many more (Docs.python.org, 2021). Hugely important for application function, using these libraries assists with maintaining, testing, and developing the program.

The project will be coded in python3 because of these strengths and will require a Linux distribution to host it. The powerful string processing will be needed for this project to handle, calculate, and modify the complex outputs that come from the tools used. Furthermore, python allows users to create their own packages for others to install. Packages such as Click and Pytest, allowing for quick and easy CLI implementation and in-development testing.

5.2. Kali Linux

Kali Linux is a powerful operating system free to download. Unlike a regular distribution like Ubuntu, it has a precise design emphasis on security, including the tools that come installed. Being able to complete various information security tasks such as penetration testing, computer forensics, and reverse engineering (kali.org, 2021). Kali has already shown potential similarly to how this project is applied regarding portability. Kali NetHunter is a penetration toolkit developed in 2014 and was designed for android mobile devices.

Many tools used in a forensic investigation, such as data carving tools, forensic image acquisition tools, and password cracking tools are installed. Kali also has the option to be deployed in multiple different ways. These include portable ARM devices, Linux Desktops, and USB devices, including the option to deploy with or without persistence. Kali will be used with the project because of its pre-installed software and it being open source. allowing for quick and easy installation on multiple types of devices, while already including the tools required for operation. This is hugely important due to everchanging development of open-source tools and Kali itself.

Though the application will be developed on Kali Linux, there is no reason why the application will not operate on different Linux distributions. Kali Linux uses Debian 10.x as its base Linux distribution, If the tools are installed alongside the necessary python3 packages, the app should operate correctly on Ubuntu for example.

5.3. Raspberry Pi 4

If the project were to include all tools from Kali, then the project would be immediately out of scope. Because of this, Forin was chosen to be developed as a mobile 'friendly' application. The Raspberry Pi 4 is a single board computer with an ARM architecture CPU.

Processor:	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
Memory:	8GB LPDDR4-3200 SDRAM
Bluetooth:	Bluetooth 5.0
Ethernet:	2.4 GHz and 5.0 GHz IEEE 802.11ac wireless
USB:	2 USB 3.0 ports; 2 USB 2.0 ports.
Multimedia:	H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode) OpenGL ES 3.0 graphics
Storage:	Micro-SD card slot for loading operating system and data storage
Input Power:	5V DC via USB-C connector (minimum 3A*)
Operating Temperature:	0 – 50 degrees C ambient

Table 4: *Raspberry Pi 4 B Specifications* (Raspberry Pi Foundation, 2021)

The Pi comes with some ports and protocols we can take advantage of. 8GB of memory is a good amount on a small device like this, with the model below only having 4GB. The low input power shows that this device could be powered by a portable battery bank, and Bluetooth is extremely secure but very short distance. However, the most important aspect and the one that arguably made this possible was the inclusion of 2 USB 3.0 ports. "USB 2.0 offers a transfer rate of 480 Mbps and USB 3.0 offers speed up to 5 Gbit/s, which is 10 times faster." (Irene, 2020) making this type of task unbearably slow on these devices until recently.

Another compelling aspect of the Pi is its modularity. Users can customise these devices to their liking, attaching GPUs for processing power or a touchscreen and battery bank for a portable device. The project will be taking advantage of this feature by changing the Pi's boot drive, using the most out of the USB 3.0 ports. Standard, the Pi uses a SD card as a boot, low storage and slow, using an SSD connected through USB can increase speeds and storage space on the device.

6. Design & Architecture

The 6 stages to digital investigation are identification, preservation, collection, examination, analysis, and presentation (Tahiri S., 2016). The scope of the project shows that only stages collection and examination are involved as the application wants to assist in usage of the tools during examination and enable users to collect corrupted and deleted data. Traditionally, a suspect device is identified and collected, taken to a lab elsewhere to analyse. Investigators will use many tools, techniques, platforms, and devices to examine data correctly and efficiently.

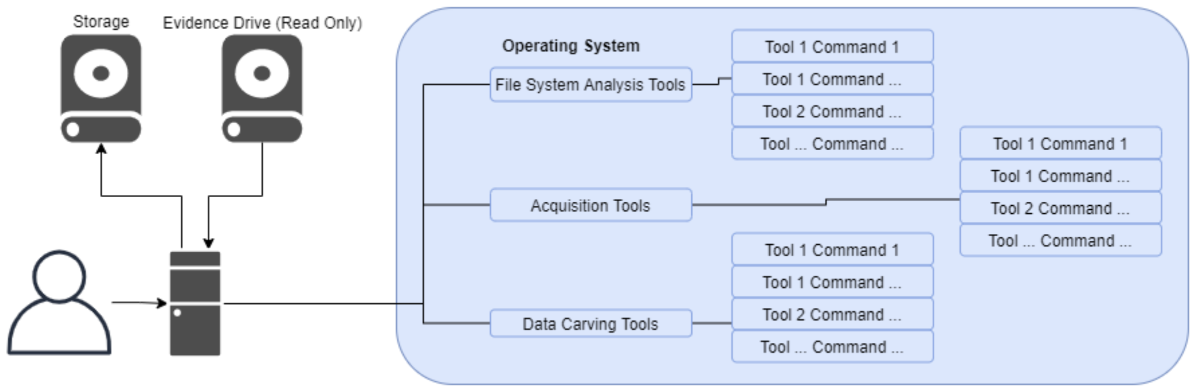


Figure 4: *Traditional deployment example*

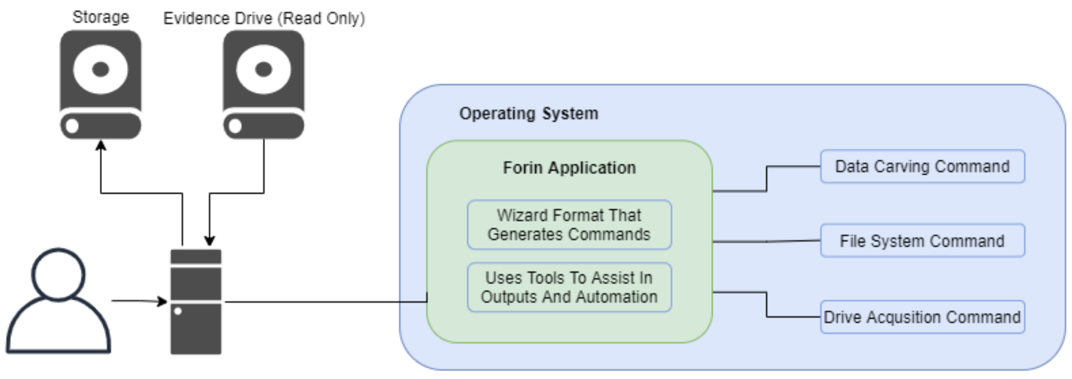

Storage
Evidence Drive (Read Only)
Operating System
Forin Application
Wizard Format That Generates Commands
Uses Tools To Assist In Outputs And Automation
Data Carving Command
File System Command
Drive Acqusition Command

Figure 5: *Forin app deployment example*

The design of Forin aims to be a wrapper that executes high privilege commands based on user input. To accomplish this, most tools work independently to one another due to different output data types and formats, meaning dependency of one another within the program should be minimal. Firstly, these tools are created and maintained by their authors, if one were to have a bug, for instance. Scalpel does not function on the Pi, so the application should still operate correctly and execute any other tool. Secondly, it is important for development of the application, if a new tool is to be added, then it can simply be added to the program without altering existing implementations. This does not count for repeated program code. Finally, Kali updates every quarter year, and in doing so the tools that are included change, as well as being different across platforms. Ideally, the application should run on any Linux distribution and allow for the tools to be installed on the operating system of choice.

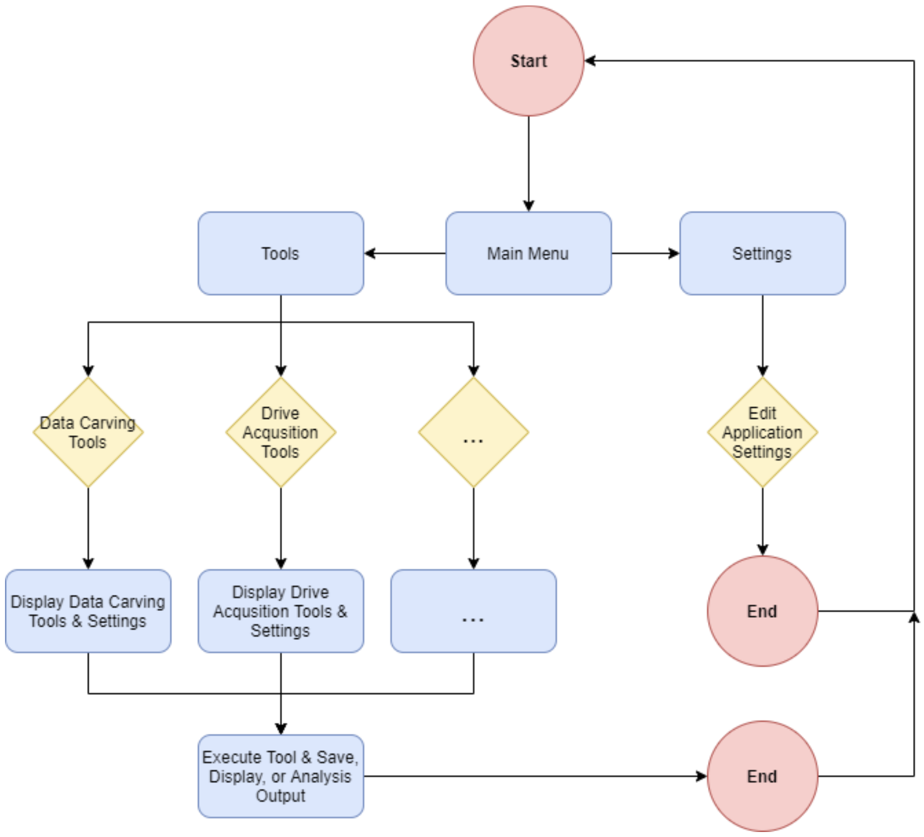

Start
Tools
Main Menu
Settings
Data Carving Tools
Drive Acqusition Tools
...
Edit Application Settings
Display Data Carving Tools & Settings
Display Drive Acqusition Tools & Settings
...
End
Execute Tool & Save, Display, or Analysis Output
End

Figure 6: *Initial Usage Diagram*

The portable aspect can be applied to the examination. With cloud services on the rise, and remote connections being more favourable with covid-19, these portable devices can be deployed on scene so an investigator can inspect data via a remote connection. Realistically, this could pose more problems than at first glance, including legal or ethical issues. Nevertheless, in a personal or non-legal environment or if the objective is mass data collection, these devices could be deployed in unique circumstances or on mass for low cost.

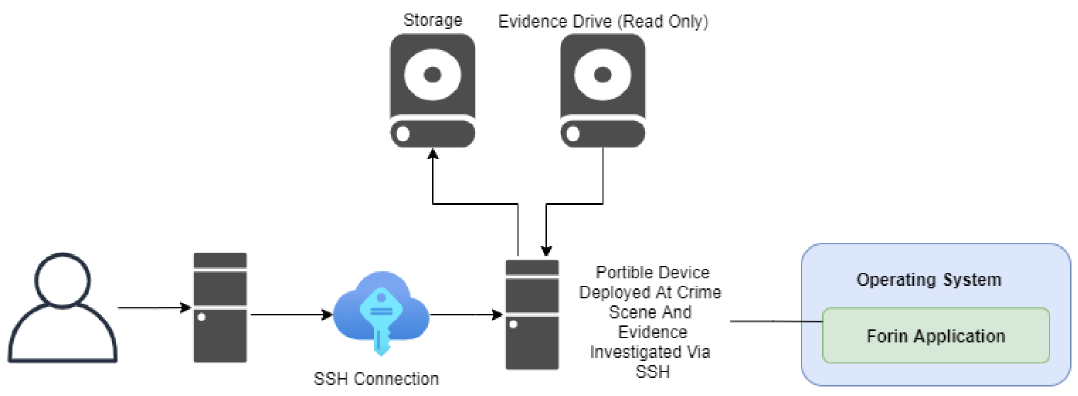

Storage
Evidence Drive (Read Only)
SSH Connection
Portible Device Deployed At Crime Scene And Evidence Investigated Via SSH
Operating System
Forin Application

Figure 7: *Forin app SSH deployment example*

As stated, Forin will be a command line application. Using already existing Linux packages and importing python packages to accomplish the required objectives while ensuring that the application is easy to use and is user friendly. Applying this rule to development ensures that it can be deployed effectively on multiple devices. If a user has deployed Forin on the PI with a touchscreen and not a keyboard, then the application must operate by either use of the touchscreen or by easy use of the onscreen keyboard.

7. Requirements

The requirements for the project include all functional and non-functional requirements. Understanding the requirements before project initiation can ensure that the application remains within the project scope. Prioritising tasks and organising them by analysing their capabilities, function, requirements, and objectives. The priority requirements tables below have a scaling of between low and urgent, urgent being a requirement for application function and MVP, or low being an additional idea. In between is important and medium, following the format of 365 planner, important being desired addition and normal being possible addition.

FORIN APPLICATION FUNCTIONAL REQUIREMENTS

Requirement	**Rating**
The user should be able to easily navigate a terminal to perform forensic tasks.	Urgent
The user should be able to execute commands for other tools to perform their objectives.	Urgent
The user should be able to use the application without hindrance (Crashes and Bugs).	Urgent
The user should be able to save settings and be persistent.	Urgent
The user should be able to save a record of any actions performed on it (Record keeping).	Important
The user should be able to set their identification or name.	Normal
The user should be able to acquire a forensic image of a drive.	Urgent

The user should be able to carve forensic data off a drive.	Urgent
The user should be able to recover deleted files.	Urgent
The user should be able to recover corrupted data.	Urgent
The user should be able to inspect and view the file system information of a forensic image.	Urgent
The user should be able to analyse PDF Files.	Urgent
The user should be able to crack passwords.	Normal
The user should be able to manage cases.	Low
The user should be able to store evidence they are trying to extract.	Urgent
The user should be able to view the evidence they have collected.	Urgent
The user should be able to use the tools easily with the assistance of automation and user inputs.	Urgent
The user should be able to compare hashes in a directory with the use of a hash file.	Normal
The user should be able deploy the application in different environments.	Important
The user should be able to set up SSH automatically through the application.	Low
The user should be able to use CLI commands for quicker usage.	Low

Table 5: *Forin Functional Requirements*

FORIN APPLICATION NON-FUNCTIONAL REQUIREMENTS	
There should be a code repository showing progress and storing backups.	Urgent
The application should be accessible to any user who wishes to receive a copy.	Urgent
There should be correct documentation for aspects of the project including final UML diagrams and licensing.	Urgent
There should be application of good development and coding practices.	Important
There should be no interference with any technologies that are not required for core function and should not be 'harmful' to the user.	Urgent
There should be the option to install and operate on multiple different platforms such as a portable device or VM.	Important

Table 6: *Forin Non-Functional Requirements*

8. Development

Development will follow the aforementioned method and design, allowing for tools to be designed, tested and implemented through different stages of development. Following these methods will ensure MVP is met, and development runs smoothly with a total development time for the entire project being 32 weeks, with a total of 16 bi-weekly sprints (Appendix 2).

8.1. Sprint 0 (Project initiation & MVP)

The first fortnight of development consisted of documentation required for pre-development. Firstly, setup of the project repository to hold any documentation developed and to record progress of the project. Next, a test virtual machine (VM) was created to test any applications that needed to be learned and to compare against the Pi device in late-stage testing.

Importantly, testing of python on Kali Linux was the first primary concern. Python should work on Kali as expected 'out of the box' having python3 installed as of version 2020.4 and understanding how the project would receive inputs from the user was the most vital usage requirement. Finally, understanding, and initial design shows how the program could be programmed. The VM will be crucial in the early stages of development as it will be used to great effect testing tools and by having persistence, as It is assumed that the Pi may encounter technical faults with usage as Kali on ARM devices.

Finally, after quick inspection of tools to be included, an MVP was also established. The MVP looked at tools in the digital investigation field and it was decided to use forensics techniques to classify each step. The tools were chosen for differing reasons including complexity, usefulness, and usability, following the requirements specified for the project, only looking at the urgent requirements.

FORIN APPLICATION MINIMUM VIABLE PRODUCT (MVP)

- Use tools to perform drive acquisition, accomplishing data collection.
- Use tools to examine and extract data on a file system.
- Use tools to perform data carving, extracting both deleted and corrupted data.
- Use tools to analyse PDF files.
- Use the Application to be able to customize application and tools settings.
- Use the application to navigate a simple CLI wizard.

Table 7: *Forin Application Minimum Viable Product*

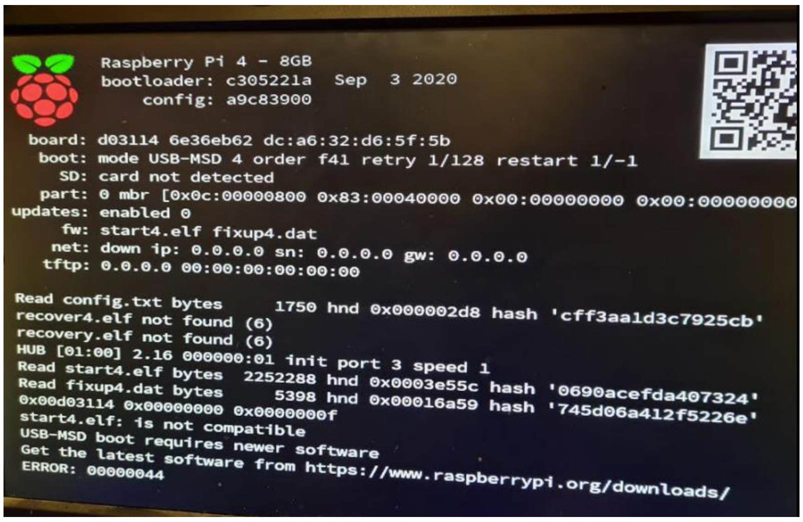
8.2. Sprint 1-5 (Early-Stage Development)

Early-stage development will include all the necessary pre-requirements for the project looking at software and hardware requirements. Firstly, all hardware required for the urgent requirements and for testing needs to be acquired and purchased. The Raspberry Pi4 will then be used as a test platform for the tools to test their effectiveness, gathering an understanding of if it would be a viable option to include or not. Furthermore, correct understanding of how these tools operate, knowing their commands, and how to format the outputs, is crucial for implementation.

The most important aspect to be covered first and foremost is understanding the capabilities of the tools on kali Linux. Looking at the operating system shows us what tools are automatically installed by default, making the tool more favourable as the user will not have to manually install tools. Furthermore, understanding which tools are used in a CLI format or knowing if their outputs can be captured without complications will limit the compatible tools greatly, determining which tools can be implemented.

Continuing, understanding how all these tools functioned was equally as important as knowing how to construct the commands, a requirement for the application. Files and directories will have to be selected by the user via terminal inputs, along with custom menus to represent certain options depending on the selected tool, desirably generating a command that executes without error and how the user specified. This posed a challenge early on as a user could input several options in the terminal and a substantial method of forcing certain inputs would have to be integrated. For early development, simple input detection was used to check for a selected number or string in specific use cases like specifying a directory.

After testing the tools and operations, multiple discoveries were made that had to be addressed. Firstly, the Pi did not perform as well as anticipated out of the box, Kali Pi V2020.4 was unstable at first, either having application crashes, read and write errors, along with overall slow performance. For a remedy, it was decided to take up a single usb3.0 port to attempt to boot the Pi4 from a USB SSD (Solid State Drive) instead of a regular SD card.

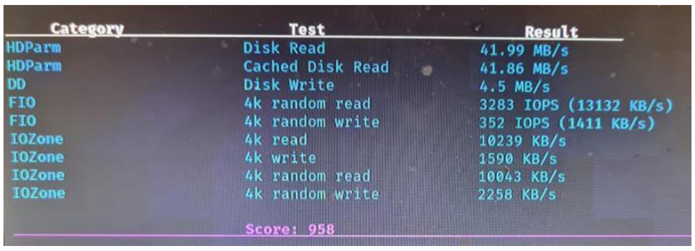


```
        Raspberry Pi 4 - 8GB
        bootloader: c305221a  Sep  3 2020
            config: a9c83900

  board: d03114 6e36eb62 dc:a6:32:d6:5f:5b
   boot: mode USB-MSD 4 order f41 retry 1/128 restart 1/-1
     SD: card not detected
   part: 0 mbr [0x0c:00000800 0x83:00040000 0x00:00000000 0x00:00000000
updates: enabled 0
     fw: start4.elf fixup4.dat
    net: down ip: 0.0.0.0 sn: 0.0.0.0 gw: 0.0.0.0
   tftp: 0.0.0.0 00:00:00:00:00:00

Read config.txt bytes     1750 hnd 0x000002d8 hash 'cff3aa1d3c7925cb'
recover4.elf not found (6)
recovery.elf not found (6)
HUB [01:00] 2.16 000000:01 init port 3 speed 1
Read start4.elf bytes  2252288 hnd 0x0003e55c hash '0690acefda407324'
Read fixup4.dat bytes     5398 hnd 0x00016a59 hash '745d06a412f5226e'
0x00d03114 0x00000000 0x0000000f
start4.elf: is not compatible
USB-MSD boot requires newer software
Get the latest software from https://www.raspberrypi.org/downloads/
ERROR: 00000044
```

Figure 8: *Raspberry Pi4 Boot Load Error Screen*

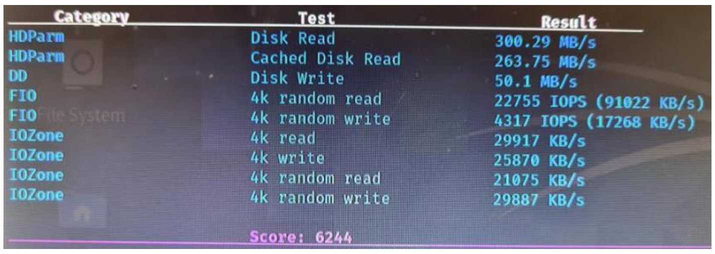
A great guide by James Chambers demonstrates how this is done by changing the partition unique ID (PATUUID) to that of the connected SSD by editing /boot/cmdline.txt (Chambers J., 2020). This was a massive improvement on performance, increasing noticeably on initial boot, with downloads installing faster and operations feeling smoother. Luckily, James also created a python script in which we can test the performance of the device and compare it to others on his website. With the PI scoring just low of 1000 with a SD card, the SSD proved to be worth the time by increasing its score to around 6000, without the addition of quirks.

Surprisingly, with this addition, not only did the Pi operate more effectively, but was also more stable. Crashing was less persistent and use of the VM reduced due to easier development and testing on the Pi. This was vital not only for development but for the viability of the project as well. Looking at James' website again, there are scores of 10,000 or more, showing that with the edition of a SSD connected, read/write speeds increase greatly thus boosting the performance. Before it could have been argued it would be too slow for everyday or professional operation.



```
     Category                   Test                   Result
HDParm                     Disk Read                41.99 MB/s
HDParm                     Cached Disk Read         41.86 MB/s
DD                         Disk Write               4.5 MB/s
FIO                        4k random read           3283 IOPS (13132 KB/s)
FIO                        4k random write          352 IOPS (1411 KB/s)
IOZone                     4k read                  10239 KB/s
IOZone                     4k write                 1590 KB/s
IOZone                     4k random read           10043 KB/s
IOZone                     4k random write          2258 KB/s

                           Score: 958
```

Figure 9: *Pi4 SD card performance*



```
     Category                   Test                   Result
HDParm                     Disk Read                300.29 MB/s
HDParm                     Cached Disk Read         263.75 MB/s
DD                         Disk Write               50.1 MB/s
FIO                        4k random read           22755 IOPS (91022 KB/s)
FIO                        4k random write          4317 IOPS (17268 KB/s)
IOZone                     4k read                  29917 KB/s
IOZone                     4k write                 25870 KB/s
IOZone                     4k random read           21075 KB/s
IOZone                     4k random write          29887 KB/s

                           Score: 6244
```

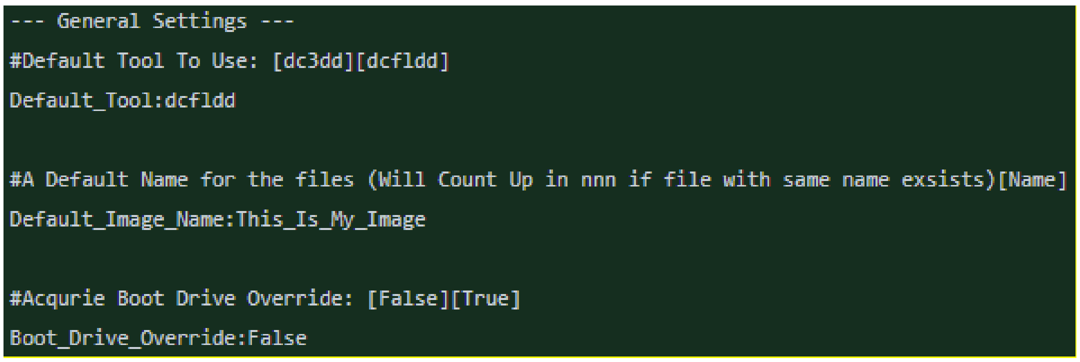
Figure 10: *Pi4 SSD performance*

8.3. Sprint 6-10 (Mid-Stage Development)

Mid stage development was the stage set primarily for early prototyping, with the expectation of starting full application development by sprint 7 at the latest. Unfortunately, this stage of development was heavily impacted at the beginning by health issues that delayed mid-stage development and was re-organised to accomplish tasks that did not require extended periods of time to complete. This was due to extended periods of work causing issues and was not diagnosed quickly due to complications and delays because of Covid-19. Nevertheless, it was decided that prototyping would have reduced time allocated while prioritizing the MVP requirements.

The first objective was testing the tools on the Pi4, a delayed objective from sprint 4, and It was found that some of the tools did not function correctly on the device. Scalpel is a data carving tool and is an adaptation of the tool Foremost. Though usage of these tools is near Identical, using a config file to choose what file types to carve, Scalpel does not function on the Pi4 at all. Luckily, early planning for this ensured that the input, outputs, and operations for Foremost were also understood and could be applied in a very similar way. As Scalpel did not work on the Pi device, it was still decided to add the tool as the application could be deployed on a device other than a Pi, giving users multiple options. Furthermore, testing the performance of the applications took place throughout mid-stage development. This was originally planned to take place in late-stage development but was pushed forward as it did not require attendance while operational and did not need the application to be functional to test the performance.

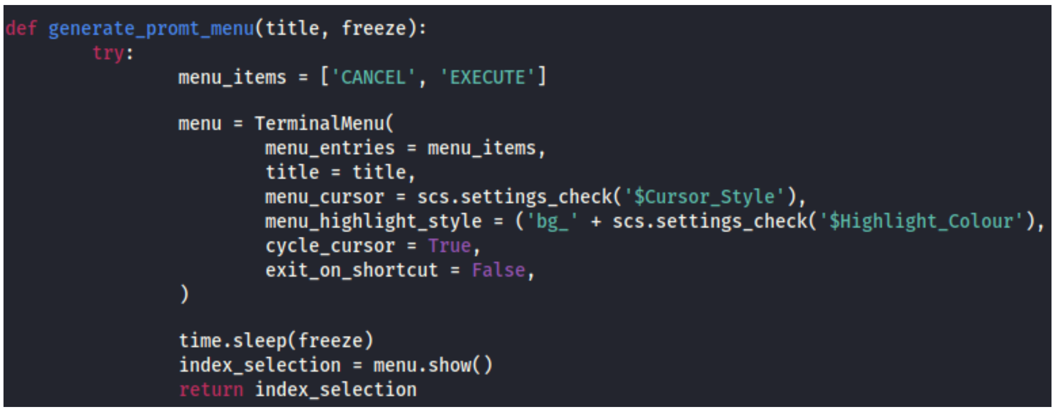
Early prototypes of the app included classes to store any information required for command generation and display. Data such as the drives path is collected via command *fdisk*, which displays all the connected drives to the user, allowing for easy identification of drives and partitions. Extracting the output to a .txt file allows for slicing of the text to extract certain data. With use of another .txt file, settings can be predetermined or changed on the fly to accompany different execution options. This is done with a simple script that checks the variable of the settings, based on a unique call.



```
--- General Settings ---
#Default Tool To Use: [dc3dd][dcfldd]
Default_Tool:dcfldd

#A Default Name for the files (Will Count Up in nnn if file with same name exsists)[Name]
Default_Image_Name:This_Is_My_Image

#Acqurie Boot Drive Override: [False][True]
Boot_Drive_Override:False
```

Figure 11: *Early Settings Config file*

The drive acquisition implementation was finished during sprint 9, taking a month to fully implement the tools. At this stage, it was clear the current method of accepting string or integer inputs would not be suitable. The high number of commands that could be accepted in the Linux terminal meant usability of the project was lacking. Users could put in several specific inputs that would either crash the program, or result in incorrect command generation, thus a solution had to be found. It was decided that Forin would take advantage of a terminal menu package to simulate a graphical user interface (GUI), using arrow keys or shortcuts to select commands. Simple_term_menu allowed the application to force specified options, removing the possibility for error by the user entirely, leaving only selection errors that could generate the wrong in the command (Meyer I., 2019). This largely solved the problem providing a good amount of flexibility and usability.



```
def generate_promt_menu(title, freeze):
        try:
                menu_items = ['CANCEL', 'EXECUTE']

                menu = TerminalMenu(
                        menu_entries = menu_items,
                        title = title,
                        menu_cursor = scs.settings_check('$Cursor_Style'),
                        menu_highlight_style = ('bg_' + scs.settings_check('$Highlight_Colour'),
                        cycle_cursor = True,
                        exit_on_shortcut = False,
                )

                time.sleep(freeze)
                index_selection = menu.show()
                return index_selection
```

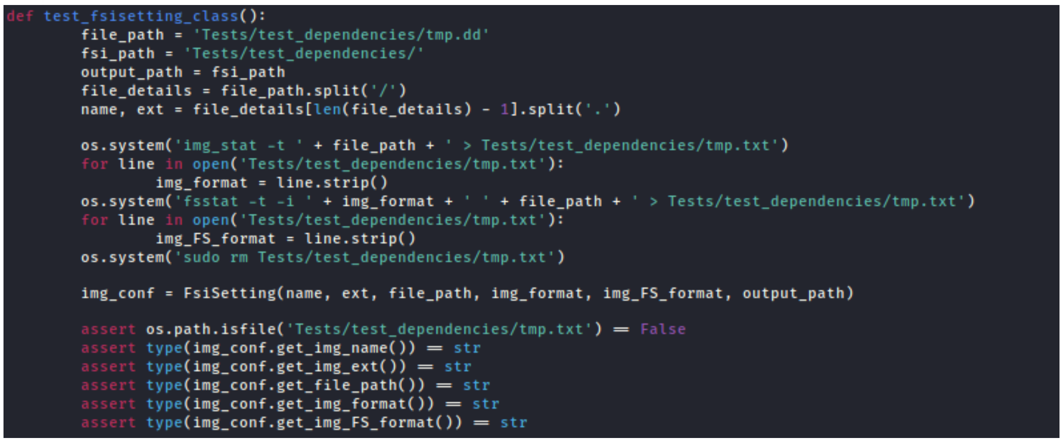
Figure 12: *TerminalMenuScript.py, generate_promt_menu function.*

Finally, the Click python package was also investigated further, only being used as an advanced print function thus far, Click can allow the user to type in full string commands and execute the program without having to 'open' the application (Ronacher A., 2014). This addition was dropped from development for two reasons, time, and usability. It was not part of the MVP so was dropped due to the problems striking development but was also argued this would not benefit enough users. If the user knows how to write these tool commands, then it could be said that they may not benefit from the application as they already know how the tools function and do not see the advantage of the application.

8.4. Sprint 11-15 (Late-Stage Development)

Late-stage development was far more successful. The time lost in the previous stages due to the circumstances was returned gracefully due to correct planning, reorganisation, and learning from previous mistakes. The objective of late-stage development was to finish the MVP detailed in planning, and to ensure that the project is high quality, following the ethos of quality over quantity. Correct testing methods will be in place to ensure that the application runs smoothly and will be followed up with any additional tools that can be added in the time remaining. Finally, end of development documentation will be created including finished UML's, while also finalizing the project with a final wave of both bug and usability testing.

The integration of a settings file means correct error handling needs to be in place to ensure that any corruptions in the settings files did not crash the application. To solve this issue, the development attempted to follow the cycle of test-driven development. "first the developer writes an (initially failing) automated test case that defines a desired improvement or new function, then produces the minimum amount of code to pass that test," (Farcic V., 2021). Tests were written before and during code design to understand how a function would complete its execution. These tests were also used to test the output of applications to ensure they gave the desired output. This was accomplished using a python package called pytest and was used to great effect, allowing for creation of assertions in separate scripts, to run tests on the developing application.

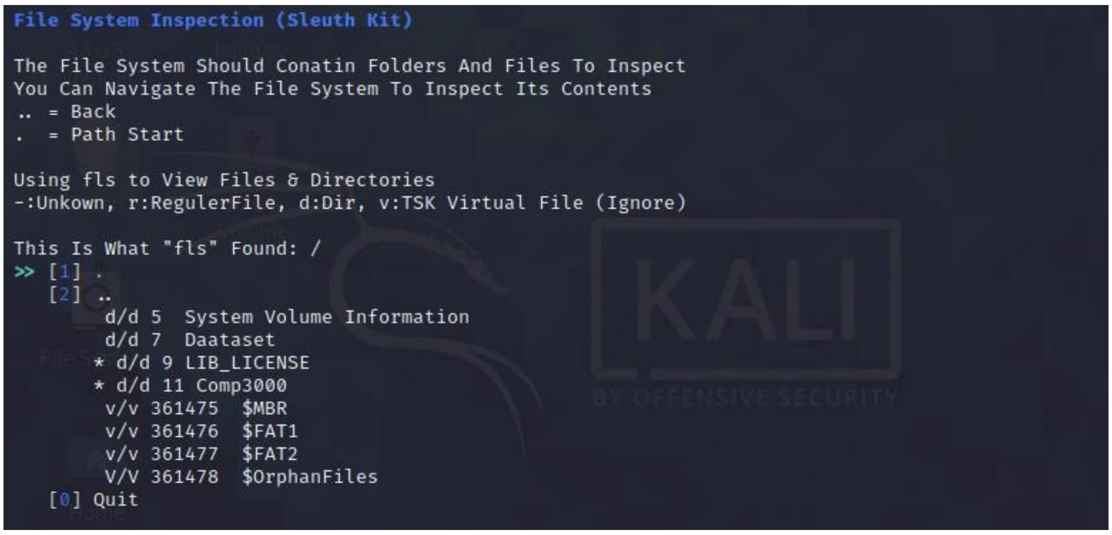


```
def test_fsisetting_class():
        file_path = 'Tests/test_dependencies/tmp.dd'
        fsi_path = 'Tests/test_dependencies/'
        output_path = fsi_path
        file_details = file_path.split('/')
        name, ext = file_details[len(file_details) - 1].split('.')

        os.system('img_stat -t ' + file_path + ' > Tests/test_dependencies/tmp.txt')
        for line in open('Tests/test_dependencies/tmp.txt'):
                img_format = line.strip()
        os.system('fsstat -t -i ' + img_format + ' ' + file_path + ' > Tests/test_dependencies/tmp.txt')
        for line in open('Tests/test_dependencies/tmp.txt'):
                img_FS_format = line.strip()
        os.system('sudo rm Tests/test_dependencies/tmp.txt')

        img_conf = FsiSetting(name, ext, file_path, img_format, img_FS_format, output_path)

        assert os.path.isfile('Tests/test_dependencies/tmp.txt') == False
        assert type(img_conf.get_img_name()) == str
        assert type(img_conf.get_img_ext()) == str
        assert type(img_conf.get_file_path()) == str
        assert type(img_conf.get_img_format()) == str
        assert type(img_conf.get_img_FS_format()) == str
```

Figure 13: *test_fsisetting_class.py, test_setting_class function*

In accordance with MVP, the next objective would be to add data carving. Foremost functions similarly to scalpel, so a single implementation could be used where selecting a tool will generate a different command and use the respective configuration file. Using simple_term_menu, the application reads the configuration file, and collects headers that the user can then select. The biggest challenge was trying to copy the format of the configuration file, as it was hard to identify what was a header selection, detecting what was and was not already enabled, while also ensuring that the new selections were saved correctly before execution. The solution was to create a script to convert all the possible headers into a list, accessing them via their unique header, and altering a code that represents enable or disable. For usability purposes, it was decided to add PhotoRec as well despite having its own CLI wizard and could be argued that creating a wrapper that simply launches a program would not be acceptable. Nevertheless, it is a rather powerful tool that is easy to use and another option for users to take and takes very little time to implement.

The next desired requirement was that of file system inspection (FSI) by taking advantage of tools in the Sleuth kit. The sleuth kit has a rather large collection of small tools that provides information about the file system and its contents. Using tools in succession with one another provides information that may be needed by another tool, a process that can be automated. Deciding how to apply these tools was the main challenge and it was decided to use a small amount to be able to inspect a file system and navigate it to get file information. using Icat and Istat allowed for collection of file system information that could then be used to select inodes, the identification number for that object in the file system. clever usage of these inodes allowed for a makeshift file system to be generated and navigated to inspect and extract files and directories.



```
File System Inspection (Sleuth Kit)

The File System Should Conatin Folders And Files To Inspect
You Can Navigate The File System To Inspect Its Contents
..  = Back
.   = Path Start

Using fls to View Files & Directories
-:Unkown, r:RegulerFile, d:Dir, v:TSK Virtual File (Ignore)

This Is What "fls" Found: /
>> [1] .
   [2] ..
        d/d 5  System Volume Information
        d/d 7  Daataset
       * d/d 9 LIB_LICENSE
       * d/d 11 Comp3000
        v/v 361475  $MBR
        v/v 361476  $FAT1
        v/v 361477  $FAT2
        V/V 361478  $OrphanFiles
   [0] Quit
```

Figure 14: *FSI, File System navigation implementation.*

Finally, to finish the MVP product, it was time to add PDF file analysis. The primary reasoning for adding PDF analysis was to explore how file analysis could operate with this type of application, not looking at singular files from this point, with PDF files themselves being a good choice of file to explore this. PDF file analysis uses PDF-Parser and PDF-ID to collect and view the ‘objects’ of that pdf, being able to view the object contents allows users to see data like metadata or even JavaScript. With some time left before final documentation and project closure, it was also decided to add Hashcat, a password cracking tool. It could be argued to be related to penetration testing rather than digital forensics, nevertheless, it was added in the hope to increase interest in the app, knowing that password cracking is more recognizable than data carving to most people.

The last sprint (15) consisted of final application testing and user testing. The application was tested for bugs rigorously before being tested by users for feedback with use of an anonymous questionnaire. The users were asked to complete a series of tasks and had varying IT experience but included Cyber Security students, Family, and friends. Their feedback allowed more bugs to be found, and some final tweaks were added such as improvement of the ‘file directory’ selection option, increasing usability. Final documentation of the application and project was also created for closure and uploaded to the GitHub repository including, usage and class UML’s and installation guides.

9. Evaluation & Testing

The application was tested at the end of development to check for any bugs and to ensure that the requirements were met. It was found that code was efficient and ran rather well with only some noticeable bugs that affected operation and were easily and quickly fixed. With usage of pytest again to run commands to check inputs and outputs of scripts, it was found that most if not all the crashes were determined by the level of corruption in the settings file. If a setting were changed manually, or a directory is deleted and cannot be found, the application would crash. Correct error handling in the scripts and classes allowed for the application to catch the crash and return the main menu instead. This was far better than the application crashing and allowed a quick response to the user to tell them what is wrong. Finally, the application was tested against the user requirements to test the overall function and operation. After testing was complete, some final UMLs were created including a usage flowchart UML for each application task and two full application class UMLs, all of which have a viewable version that can be found on the GitHub repository (Appendix 4).

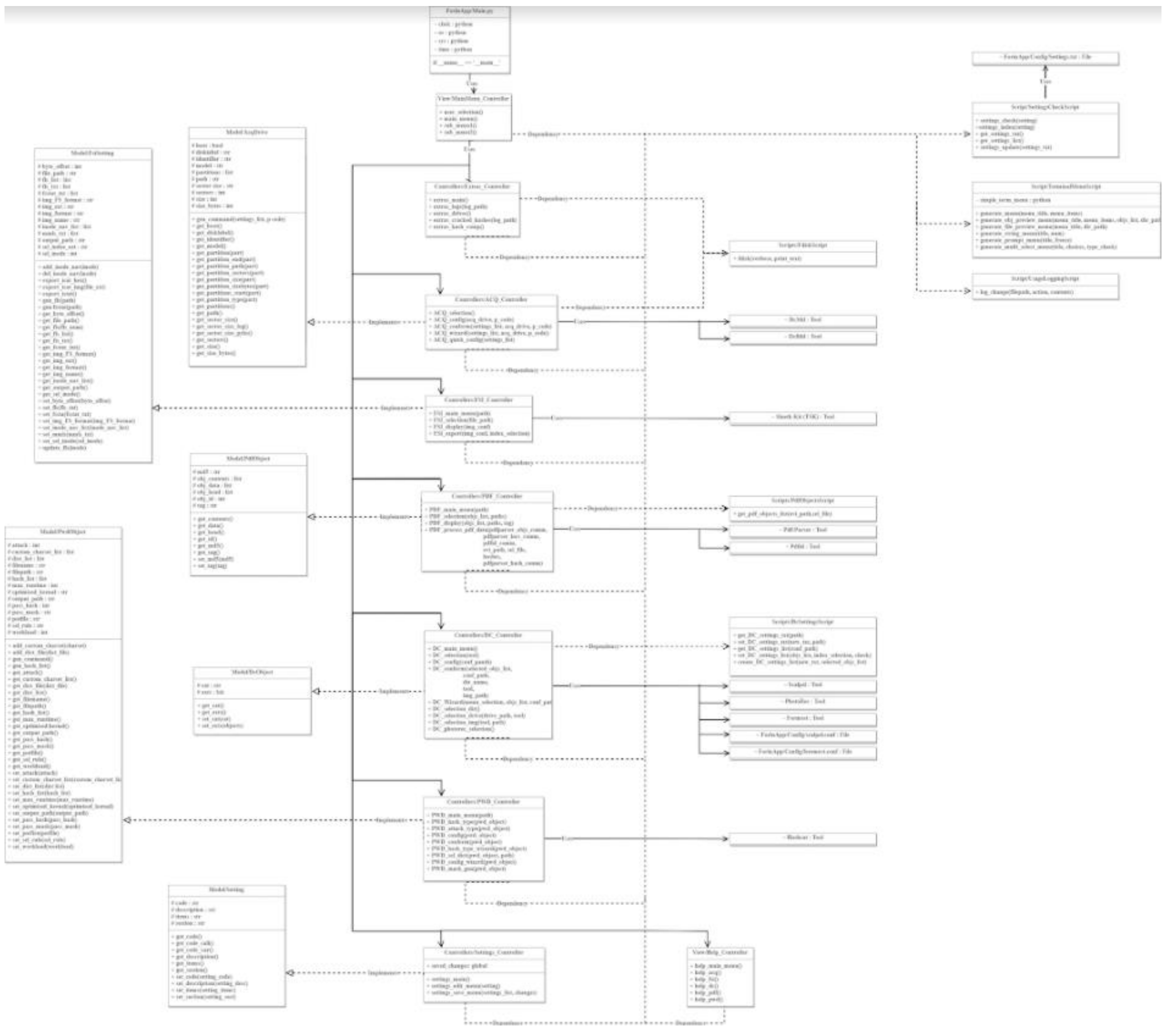


Figure 15: Forin Application UML Diagram

The usability testing element took place at the beginning of the final sprint. In total, only 4 users inspected the application alongside the developer to complete a series of tasks based on the requirements of the project. Each task allowed the user to gather an understanding of the domain, tools, and reasoning, while also trying to display why it would be useful to them. It was estimated that password cracking would be the hardest to learn while also being the most interesting, in turn hoping this would help them remember the tool for when they need it. It was also estimated that users who are unfamiliar with the domain are more likely to want to use it instead of forensic experts. The reasoning being that a user may be looking for the quick option whereas the expert may be looking for a more complex implementation. Nevertheless, each test took no longer than one hour and each of the results would provide vital information (Appendix 5).

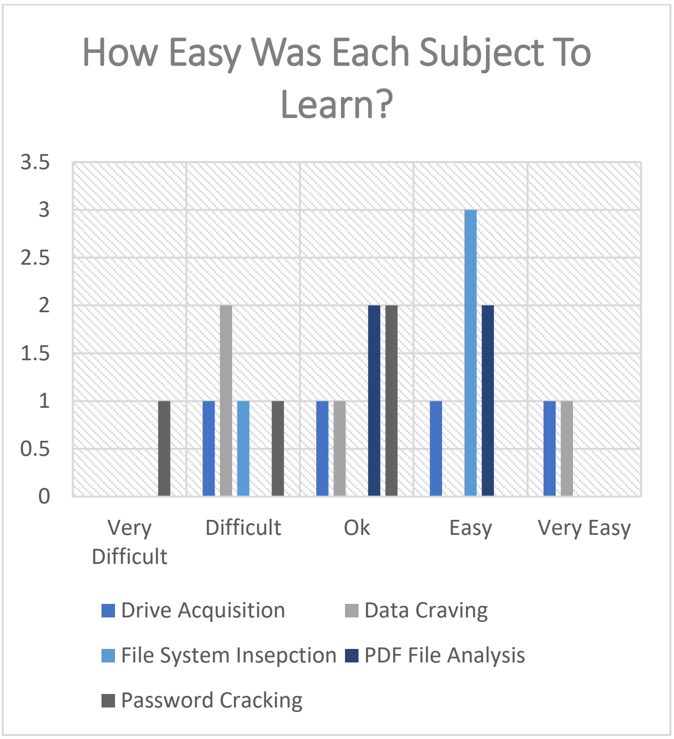


Figure 16: *Forin Questionnaire Question 1 Results*

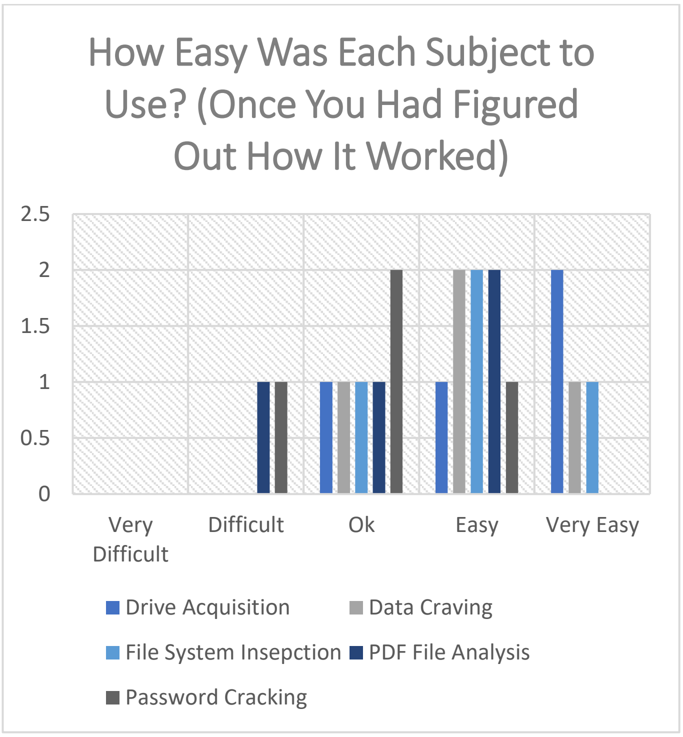


Figure 17: *Forin Questionnaire Question 2 Results*

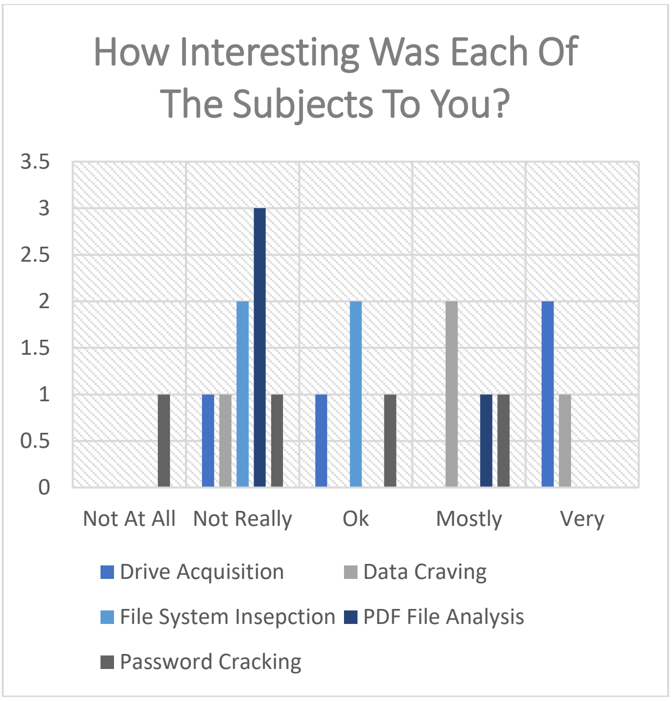


Figure 18: *Forin Questionnaire Question 3 Results*

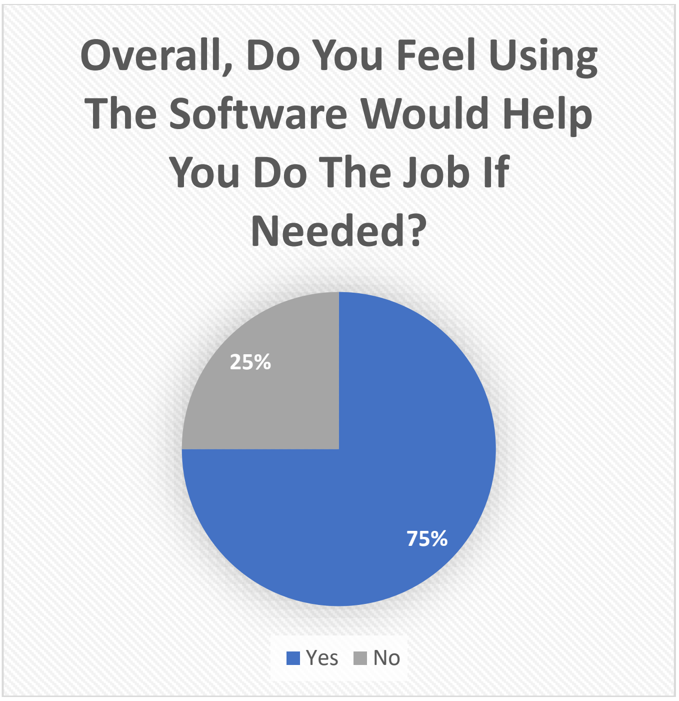


Figure 19: *Forin Questionnaire Question 4 Results*

Unfortunately, such a small sample size does not give us the ability to analyse the results accurately and rigorously. Nevertheless, there are some good and surprising results that can be discussed. Firstly, it is extremely pleasing to see that overall results for question 2 are above 'Ok', as development of the project had a large emphasis on usability. Secondly, it is surprising that Hashcat did not provide the expected interest in the application. It was assumed that password cracking would be rather interesting to users and was the primary reason for implementation. Finally, 3 out or the 4 users stated that overall, they would use the application to assist in digital forensic activities, showing some success for the application. This could be largely because it is simple to use but is not extremely complex, and so one tester said they would not as they knew of grater alternatives.

The final evaluation was to assess the overall viability of these tools in a portable environment. It is believed that it can be deployed in this manner by may require further implementation while assessing the current state of both portable devices and organizational shifts. The testing took place earlier in development due to circumstances previously explained and showed the viability of the tools that were to be implemented. Testing took place across 2 devices, the raspberry pi4 and a VM with 8 GBs of RAM and 2 CPU cores @ 3.50GHz, unfortunately GPU passthrough was not available with the VM.

DRIVE ACQUISITION TESTING RESULTS

Port	**device**	**tool**	**Disk Image Size (G)**	**Time Elapsed (s)**	**Avg. R/W Speed**
Usb2.0	Pi 4	Dc3dd	15 G	729 s	21 M/s
Usb3.0	Pi 4	Dc3dd	15 G	454 s	33 M/s
Usb2.0	VM	Dc3dd	15 G	671	23 M/s
Usb3.0	VM	Dc3dd	15 G	313s	48 M/s

Table 8: *Drive Acquisition Testing Results*

DATA CARVING TEST RESULTS

Device	**Tool**	**Disk Image File Size (Mb)**	**Number of Hits**	**Time Elapsed (s)**
Pi 4	Foremost	500	97	59
VM	Foremost	500	95	37

Table 9: *Data Craving Test Results*

HASH CRACKING TEST RESULTS

Device	**Tool**	**Mask used**	**Total No. of calculated hashes**	**Estimated Completion Time**

Pi 4	Hashcat	?a?a?a?a?a?a?a?a (8)	8,840,730,624 (billion)	14 years, 164 days
VM	Hashcat	?a?a?a?a?a?a?a?a (8)	5,8386,282,496 (billion)	2 years, 65 days

Table 10: *Hash Cracking Test Results*

The results of the tools can show the overall performance of the Pi and evidently shows that the device can perform better in areas compared to others. Drive acquisition was used to test the read and write speeds of the device, though this can be largely dependent on the media being acquired. The results show that the average speed for USB 2.0 across devices was similar and the outdated peripheral reached its read/write limits. Using USB 3.0, the VM had a 50% increase in performance and can be explained both by the lower performance CPU and a noticeable 'hanging' after extended use. For data carving, the Pi took twice the time it took the VM showing that data craving could be argued to be unviable for these devices. If the drive were 2TB for example, it would take the pi around 2000 seconds (33 minutes) to complete, whereas a VM would only take 1200 (20 Minutes), with a top of the range device being even quicker. Finally, Hashcat was tested and to no surprise performed appallingly, taking longer than the VM by 700% or 12 years. The Pi was not designed for fast math calculations however, its modularity could benefit users by being a cheap device to take advantage of older GPU equipment that has been replaced and updated. This testing both showed what is and is not viable for the Pi, it shows that drive acquisition is viable by applying a mass deployment approach. capturing many drives at once with multiple Pi's could be a viable strategy, and with use of the application could help speed up the process by using the pre-applied settings. While also showing that tools such as data craving and hash cracking, could struggle to perform without specific circumstances.

10. Project End-Report

Overall, it is believed that the project has been a success, completing all urgent requirements following an MVP and agile approach while also having released a software application to assist users in digital forensics. Some aspects of the application were not perfect, and the project does show some flaws in the viability, but this type of application and principle does present benefits. These goals were reached with constant planning, designing, and testing, while implementing digital forensics in a CLI environment. It was designed to allow for implementation on many frameworks and functions to a high standard, containing minimal bugs and robust error handling.

10.1. Requirements & Objectives

The Application meets most of the requirements that were originally set. The requirements specify that the user should be able to collect and analyse complete, deleted, and corrupted data from a hard drive and use forensic tools to perform these tasks. These objectives have been completed and the attempt to apply these tools in a way that is beneficial to both novices and experts alike posed a challenge through development. None of the tools had full implementation of options and settings to allow for more stages, instead of a complex implementation of a single tool. With this, it was also believed to add multiple tools for each stage instead of a complex implementation of a single tool due to accessibility. This also makes sense in a forensic environment because each of the tools can produce different results when looking at data. meaning investigators may use multiple tools regardless due to some tools producing data that may be inaccessible by another tool accomplishing the same job. Accessibility was a major contributor to the project and due to this challenge, it is believed to have been applied appropriately due to operational issues with some of the tools and giving the user more options to complete their tasks.

Furthermore, the objectives specify that the user must be able to set and save settings. This was accomplished with the simple use of a .txt file and was implemented to a high standard. Any settings that can be specified with the tools can usually be pre-determined using the settings file, allowing for quick investigation, and even preparing for errors when editing the file manually. Some tools took advantage of this greatly while others only required a small amount. Continuing, it also introduced settings for the application to allow the user to change some aspects of a tool to their liking during operation while also containing options for the application to assist in file system navigation for example. The settings, along with usage of the tools was also recorded with the use of a logging system. The implementation was a simple file output and is just a record of usage, a small addition for investigations to assist in the chain of command.

However, not all the requirements were met. The current implementation of the tools is fine for its current use but could be easily argued that some tools could have had better implementation. For example, drive acquisition uses two of the most common tools and allows for almost all their settings to be selected whereas PDF file analysis just allows you to view the objects of a PDF file and not much else. Additionally, some of the low rated requirements were dropped completely due to the circumstances interrupting development. It would have been nice and a huge boost to usability if SSH could have been set up with the click of a button, helping novice users greatly but was dropped due to its low priority.

The final major requirement was ensuring the application was easy to use while also educating new users and assisting investigators. It is believed that this is accomplished based on the results from the project's usability testing. Most of the answers from these results suggest that a variety of users will primarily find this application useful in one way or another, with the majority explaining that they would use the application for the tasks and showing they found the application was easy to use. However, it is unsure how helpful this could be to expert investigators, due to current workflows and in house scripts already being implemented and it is estimated that future implantation is needed to achieve a noticeable impact.

10.2. Application Viability

The viability of this type of project in the real world is questionable as previously suggested. Firstly, not all the tools are appropriate and can be seen by the test results in the previous section. Drive acquisition as stated could be deployed on mass, but password cracking should be avoided. However, the tests did not include file system analysis nor PDF analysis primarily because they functioned quickly in both the application and one their own. This shows that perhaps these portable devices are more suited to only analysing data instead of collecting it, and if this were truly the case, then it could be suggested that a GUI would be the better option. The option of a GUI could allow it to be more accessible to users at the expense of deployment options, while perhaps costing more money to develop and implement.

Nevertheless, it is believed that a tool like this application would be viable in today's current IT climate. Everyday, the number of attacks of users increase, the number of services online increase, and the number of users online increase, meaning the risk of cyber attacks and the damage they cause in turn increases. This suggests that users should understand how to recover and retrieve data after an attack, following a principle like; if a user's car has a flat, they should know how to change the tire. Furthermore, devices like the Pi are extremely cost effective and can easily be deployed while at the same time could easily be deployed on already existing or free platforms. Allowing users to conduct some of the primary aspects of digital investigation.

11. Project Post-Mortem

The aim of this project was to test the viability of traditional forensics tools and techniques on portable devices and was accomplished with the creation of an easy-to-use CLI application to accomplish these tasks. The application delivers on this requirement alongside any urgent requirements specified, however this is not without fault. Due to circumstances, some aspects of the application were not implemented, and corners were cut to save time.

11.1 Project Management & Approach

The biggest criticism of this project would be that of its project management. Through the application development cycle, consistent and relevant commits were posted to GitHub, alongside correct sprint management during early and late-stage development on the project planner, but it is questionable during mid-stage development. Unfortunately, time had to be saved during development due to a physical medical issue, to allow for increased development time due to a daily time constraint. The absence of sprint reviews is due to incompletion after early-stage development. Though not taking a long time to complete, the reorganisation and changing circumstances made it difficult to know what could be done on a bi-weekly basis as these circumstances changed the best course of action. Furthermore, this led to a noticeable absence of the biweekly meetings, these meetings were in the later afternoon meaning attendance was not entirely possible as rest would be needed instead. It was decided to bring later objectives forward, such as performance testing of tools as they were easier to maintain alongside less prototyping to ensure enough time for full development. Notably however, most aspects of the application were completed to a reasonably high standard. This is due to the realistic MVP approach applied to development allowing for flexible design and implementation alongside an agile development cycle.

11.2 Application Code & Design

Deciding to prototype less was not a decision that was taken lightly but was required and took a small toll on development. The code of the project attempts to follow good coding standards throughout development and it is believed that this has been applied in most areas including commenting, indention, and efficiency. However, the applied object-oriented design may not be to the highest standard. Some of the python scripts should be ported to the classes themselves and in some cases, classes where not used to the highest effect. Version 0.2 of the application was built upon the prototypes created which led to early and messy code being implemented and time was redesigned later in the project to create a more appropriate implementation. Despite this, the application accomplishes its requirements with good and well applied error handling, functioning on multiple Linux platforms.

11.3 Developer Performance

Despite inconveniences through development, and with the full project being developed with covid-19 restrictions, this project hopes to show that the developer performed well. Obvious criticisms can be and have been made about the project while also highlighting what went well. A lot of new technologies were studied and tested throughout development to ensure the correct tools were chosen for the requirements alongside unfamiliar programming techniques such as test-driven development. Overall development of the project went well through early and late-stage development and is demonstrated through regular commits and updates to both the repository and planner. Correct reorganisation of tasks and priorities ensured that the project has been released to an acceptable standard that can be released in the public domain, following MVP throughout development. However, better attention to planning could have been the solution to some of the problems encountered throughout the project. Nevertheless, each problem encountered was also dealt with swiftly and appropriately. It was also a priority to complete all objectives of the application before the deadline and ensured the repository contained all necessary documentation.

11.4 Future Developments

There are a few future development options that could be followed with the first being advanced implementation of what already exists. As stated, the application does not apply every option for every tool, thus this could be implemented for greater usage and control. This would arguably be the best future implementation as it sticks within the realm of traditional forensics. This would also give a greater reason to investigators to use the application. Alongside this, other functionality options such as the click package scripts to assist experts. Furthermore, extras that were dropped from the requirements could also be implemented such as automatic SSH configuration, assisting users even further.

Other future developments could include looking at different fields of digital investigation such as live and network forensics. Though crucial to any investigation, just like traditional there are many tools that could be used and complex implementation of these could seriously extend the length of the project, making it unrealistic. So much so it could also be argued that they should have their own application. On the contrary, live forensics could be applied very well in a portable format, applying a ‘plug & go’ strategy, as the biggest problem with this field is losing the volatile data before collection.

12. Conclusions

In conclusion, this project demonstrated that digital forensics tasks can be performed and executed by novice individuals and there is the chance for improvement in the investigation process. This was done with the completion of Forin which assists investigators and novices alike, while also applying a portable model to design. The benefits of this type of application are shown to be greater for new users when compared to experts with perhaps a more advanced implementation is required to make an impact. Furthermore, the portable testing results, alongside being developed for the Pi device, shows that this could be viable in multiple different areas through mass deployment or remote connection. though admittedly these deployment options are both circumstantial and dependent. Unfortunately, what is known is that the Pi would not be viable out of the box and the device should have solid state storage at the minimum.

Finally, the success of the project is largely due to the agile based approach in which it allowed for easy and quick adjustments mid development. This allowed for reorganisation of objectives and ended in the creation of a suitable final product. Many lessons have been learned from this project with a large emphasis on project planning. The circumstances and how well they were tackled both provided crucial learning experiences. Furthermore, large amounts were learned from researching and practising new software technologies including all the tools used in the application, to understand better implementations of coding standards. Overall, the full software develop cycle has been followed, and led to the release of a functional CLI digital forensic tool.

VI. References

2013. POLICE AND CRIMINAL EVIDENCE ACT 1984 (PACE) CODE B. [ebook] gov.uk, p.16. Available at: <https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/903811/pace-code-b-2013.pdf> [Accessed 2 May 2021].

Adelstein, F., 2006. *Live forensics: diagnosing your system without killing it first: Communications of the ACM: Vol 49, No 2*. [online] Dl.acm.org. Available at: <https://dl.acm.org/doi/10.1145/1113034.1113070> [Accessed 29 April 2021].

Apache.org. 2004. Apache License, Version 2.0. [online] Available at: <https://www.apache.org/licenses/LICENSE-2.0> [Accessed 2 May 2021].

Bit Bucket. n.d. What is version control | Atlassian Git Tutorial. [online] Available at: <https://www.atlassian.com/git/tutorials/what-is-version-control> [Accessed 2 May 2021].

Business Model Toolbox. n.d. Minimum Viable Product (MVP) - Business Model Toolbox. [online] Available at: <https://bmtoolbox.net/tools/minimum-viable-product/> [Accessed 2 May 2021].

Chambers, J., 2020. *Raspberry Pi 4 USB Boot Config Guide for SSD / Flash Drives*. [online] James A. Chambers. Available at: <https://jamesachambers.com/raspberry-pi-4-usb-boot-config-guide-for-ssd-flash-drives> [Accessed 5 May 2021].

Docs.python.org. 2021. General Python FAQ — Python 3.9.5 documentation. [online] Available at: <https://docs.python.org/3/faq/general.html#what-is-python> [Accessed 2 May 2021].

Farcic, V., 2021. *Test Driven Development (TDD): Example Walkthrough*. [online] Technology Conversations. Available at: <https://technologyconversations.com/2013/12/20/test-driven-development-tdd-example-walkthrough/> [Accessed 10 May 2021].

Fruhlinger, J., 2019. *What is digital forensics? And how to land a job in this hot field*. [online] CSO Online. Available at: <https://www.csoonline.com/article/3334396/what-is-digital-forensics-and-how-to-land-a-job-in-this-hot-field.html> [Accessed 26 April 2021].

Hazevytch, L. and Vilchynska, H., 2020. Agile Advantages For Software Development | DevCom. [online] DevCom. Available at: <https://devcom.com/tech-blog/agile-advantages-for-business/> [Accessed 1 May 2021].

Jonpeddie.com. 2017. *Overall GPU shipments increased 9.3% from last quarter, AMD increased 8% Nvidia increased 30%*. [online] Available at: <https://www.jonpeddie.com/press-releases/overall-gpu-shipments-increased-9.3-from-last-quarter-amd-increased-8-nvidi> [Accessed 26 April 2021].

Kali.org. 2021. The Most Advanced Penetration Testing Distribution. [online] Available at: <https://www.kali.org/> [Accessed 2 May 2021].

Legislation.gov.uk. 1990. Computer Misuse Act 1990. [online] Available at: <https://www.legislation.gov.uk/ukpga/1990/18/section/1> [Accessed 1 May 2021].

Meyer, I., 2019. *simple-term-menu*. [online] PyPI. Available at: <https://pypi.org/project/simple-term-menu/> [Accessed 7 May 2021].

MiniTool. 2020. USB 2.0 vs. 3.0: What's the Difference and Which One Is Better. [online] Available at: <https://www.partitionwizard.com/partitionmagic/usb-2-0-vs-usb-3-0-006.html> [Accessed 2 May 2021].

Opensource.org. n.d. The MIT License. [online] Available at: <https://opensource.org/licenses/MIT> [Accessed 1 May 2021].

Purplesec.us. 2019. 2019 Cyber Security Statistics. [online] Available at: <https://purplesec.us/resources/cyber-security-statistics/#:~:text=In%202018%20there%20were%2080%2C000,and%20health%20records%20left%20unprotected> [Accessed 29 April 2021].

Putman, P., n.d. Script Kiddie: Unskilled Amateur or Dangerous Hackers? | United States Cybersecurity Magazine. [online] United States Cybersecurity Magazine. Available at: <https://www.uscybersecurity.net/script-kiddie/> [Accessed 1 May 2021].

Raspberrypi.org. 2021. Raspberry Pi 4 Tech Specs. [online] Available at: <https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/> [Accessed 2 May 2021].

Ronacher, A., 2014. *click*. [online] PyPI. Available at: <https://pypi.org/project/click/> [Accessed 7 May 2021].

Rossum, G., 2021. History and License — Python 3.9.5 documentation. [online] Docs.python.org. Available at: <https://docs.python.org/3/license.html> [Accessed 1 May 2021].

Steube, J. and Gristina, G., n.d. hashcat wiki. [online] Hashcat.net. Available at: <https://hashcat.net/wiki/doku.php?id=hashcat> [Accessed 29 April 2021].

Tahiri, S., 2016. *Digital Forensics Models - Infosec Resources*. [online] Infosec Resources. Available at: <https://resources.infosecinstitute.com/topic/digital-forensics-models/#:~:text=This%20model%20was%20the%20base,some%20candidate%20techniques%20or%20methods> [Accessed 4 May 2021].

VII. Appendices

Appendix 1: Installation & Usage Guide

Forin Application Installation & Usage

All files can be found on the GitHub Repository.

Installation:

Download Virtual Machine (Recommended: VM Workstation Player v16 64-bit Windows)

Link: https://my.vmware.com/en/web/vmware/downloads/details?downloadGroup=WKST-PLAYER-1611&productId=1039&rPId=63655

Download Kali Linux 64-bit (Live)

Link: https://www.kali.org/downloads/

Setup Virtual Machine and Install Kali Linux (Using Workstation Player):

- Go To: Player > File > New Virtual Machine...
- Select Installer disc image file (iso) and select the Kali Live .iso image that was downloaded, Next.
- Select Linux (Debian 10.x 64-bit), Next.
- Specify a name and save directory, Next.
- Select the disk space you wish to set (recommended 20GB), Next.
- Select Finish.

This .iso *IS NOT* persistent. Please suspend the VM as shutting down will reset the state of the .iso back to default, losing all data.

- Once booted select 'Live (amd64)'
- Open terminal by right clicking the desktop and selecting 'Open Terminal Here '.
- Clone the GitHub with the following command:
  - *git clone https://github.com/Jiskey/Raspberry-Pi4-Forensics*
  - Note: do not use 'sudo' here as you may from problems navigating the folder through the GUI. (super user protection)
  - Note: This could also take a while as the repo contains 50Mb of test files.
- Navigate to Raspberry-Pi4-Forensics/ForinApp/ in the terminal.
  - *cd Rasperry-Pi4-Forensics/ForinApp*
- Run the Python 'build.py' file (all python commands are run in python3).
  - *sudo python3 build.py*
  - Note: Some tools are no longer installed as of Kali v2021.1. because of this, it could take a while to complete.
- Run the Python Application.
  - *sudo python3 Main.py*

Trouble Shooting:

Though the app can work on other distributions of Linux, Kali has most of the tools installed and is a much easier deployment option. The 'build.py' file contains the commands to install the required packages and tools (refer to that file for command list), but still requires python3 to run. Kali comes with most of the packages and tools preinstalled including pyhton3 but if not, you can install it with the following command.

- *sudo apt-get install python3.7*

Usage:

Flowcharts of how the application is used can be found on GitHub under (/Docs/Usage FlowCharts):

Link: https://github.com/Jiskey/Raspberry-Pi4-Forensics/tree/main/Docs/Usage%20FlowCharts

Screenshots of the application can be found on GitHub (/Docs/Application Screenshots):

Link: https://github.com/Jiskey/Raspberry-Pi4-Forensics/tree/main/Docs/Application%20Screenshots

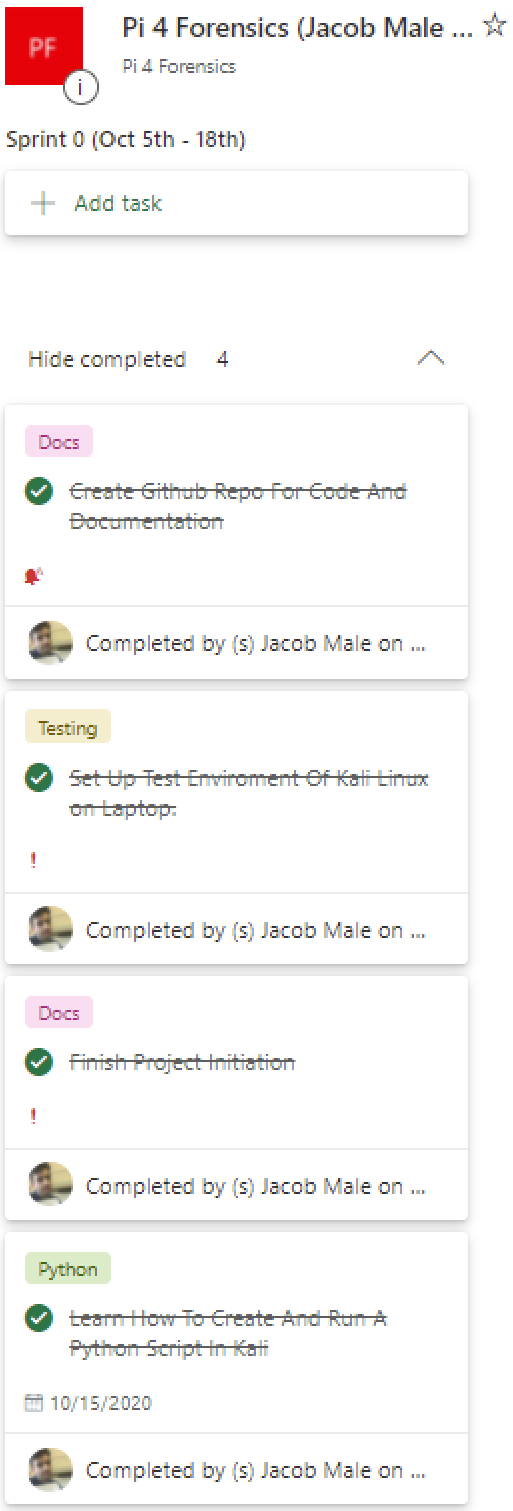
For help on what each of the tools do, you can refer to the help page within the application.

Test Files:

Test files unfortunately due to their size cannot be uploaded to GitHub. Because of this, google drive will have to be used.

Google drive link:
https://drive.google.com/drive/folders/1oRPjlfK6eSkUVrn9P_bo57VEbHDV1lm3?usp=sharing

Appendix 2: Planner & Sprints



PF

Pi 4 Forensics (Jacob Male ... ☆

Pi 4 Forensics

Sprint 0 (Oct 5th - 18th)

+ Add task

Hide completed 4

Docs

✓ ~~Create Github Repo For Code And Documentation~~

Completed by (s) Jacob Male on ...

Testing

✓ ~~Set Up Test Enviroment Of Kali Linux on Laptop.~~

!

Completed by (s) Jacob Male on ...

Docs

✓ ~~Finish Project Initiation~~

!

Completed by (s) Jacob Male on ...

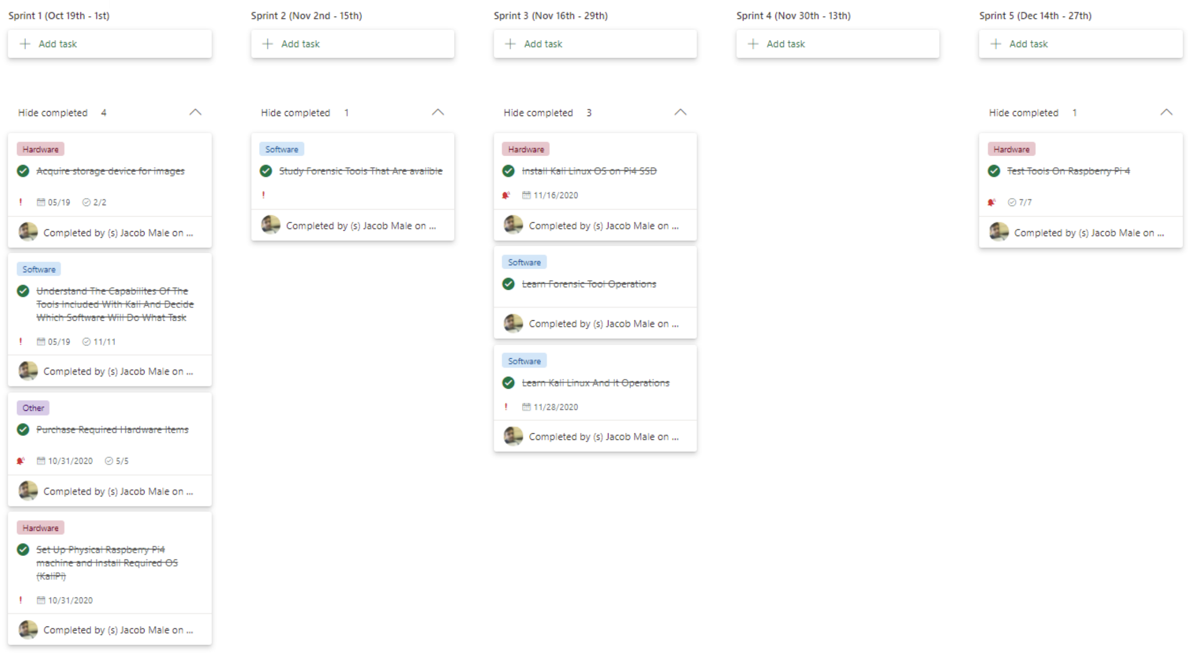
Python

✓ ~~Learn How To Create And Run A Python Script In Kali~~

10/15/2020

Completed by (s) Jacob Male on ...

Appendix 2.1: *Sprint 0 (Project Initiation)*



Sprint 1 (Oct 19th - 1st)

Add task

Hide completed 4

Hardware
~~Acquire storage device for images~~
! 05/19 2/2
Completed by (s) Jacob Male on ...

Software
~~Understand The Capabilites Of The Tools Included With Kali And Decide Which Software Will Do What Task~~
! 05/19 11/11
Completed by (s) Jacob Male on ...

Other
~~Purchase Required Hardware Items~~
10/31/2020 5/5
Completed by (s) Jacob Male on ...

Hardware
~~Set Up Physical Raspberry Pi4 machine and Install Required OS (KaliPi)~~
! 10/31/2020
Completed by (s) Jacob Male on ...

Sprint 2 (Nov 2nd - 15th)

Add task

Hide completed 1

Software
~~Study Forensic Tools That Are avalible~~
!
Completed by (s) Jacob Male on ...

Sprint 3 (Nov 16th - 29th)

Add task

Hide completed 3

Hardware
~~Install Kali Linux OS on Pi4 SSD~~
11/16/2020
Completed by (s) Jacob Male on ...

Software
~~Learn Forensic Tool Operations~~
Completed by (s) Jacob Male on ...

Software
~~Learn Kali Linux And It Operations~~
! 11/28/2020
Completed by (s) Jacob Male on ...

Sprint 4 (Nov 30th - 13th)

Add task

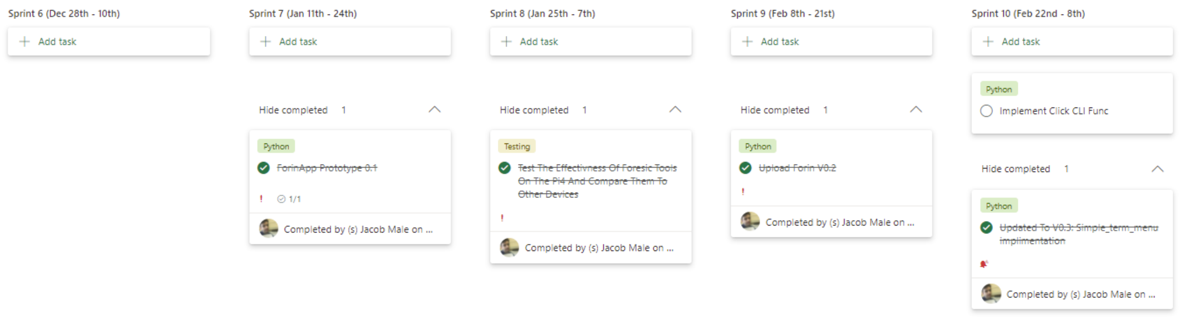
Sprint 5 (Dec 14th - 27th)

Add task

Hide completed 1

Hardware
~~Test Tools On Raspberry Pi 4~~
7/7
Completed by (s) Jacob Male on ...

Appendix 2.2: *Sprint 1-5 (Early-Stage Development)*



Sprint 6 (Dec 28th - 10th)

Add task

Sprint 7 (Jan 11th - 24th)

Add task

Hide completed 1

Python
~~ForinApp Prototype 0.1~~
! 1/1
Completed by (s) Jacob Male on ...

Sprint 8 (Jan 25th - 7th)

Add task

Hide completed 1

Testing
~~Test The Effectivness Of Foresic Tools On The Pi4 And Compare Them To Other Devices~~
!
Completed by (s) Jacob Male on ...

Sprint 9 (Feb 8th - 21st)

Add task

Hide completed 1

Python
~~Upload Forin V0.2~~
!
Completed by (s) Jacob Male on ...

Sprint 10 (Feb 22nd - 8th)

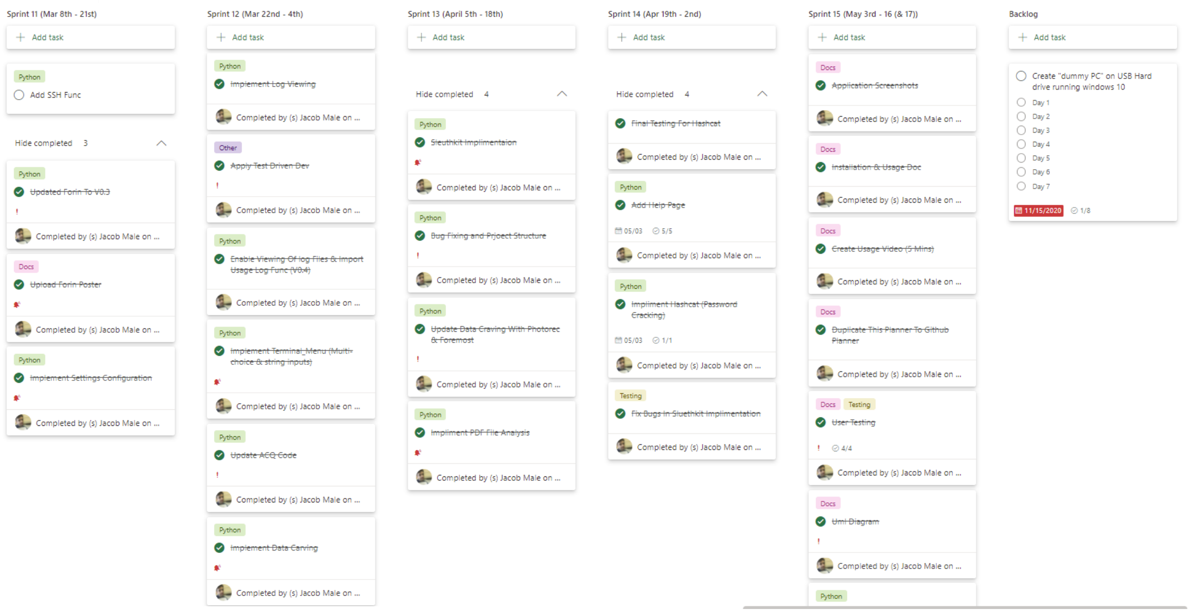
Add task

Python
Implement Click CLI Func

Hide completed 1

Python
~~Updated To V0.3: Simple_term_menu implimentation~~
Completed by (s) Jacob Male on ...

Appendix 2.3: *Sprint 6-10 (Mid-Stage Development)*



Sprint 11 (Mar 8th - 21st)

+ Add task

Python
Add SSH Func

Hide completed 3

Python
~~Updated Forin To V0.3~~
Completed by (s) Jacob Male on ...

Docs
~~Upload Forin Poster~~
Completed by (s) Jacob Male on ...

Python
~~Implement Settings Configuration~~
Completed by (s) Jacob Male on ...

Sprint 12 (Mar 22nd - 4th)

+ Add task

Python
~~Implement Log Viewing~~
Completed by (s) Jacob Male on ...

Other
~~Apply Test Driven Dev~~
Completed by (s) Jacob Male on ...

Python
~~Enable Viewing Of log Files & Import Usage Log Func (V0.4)~~
Completed by (s) Jacob Male on ...

Python
~~Implement Terminal_Menu (Multi-choice & string inputs)~~
Completed by (s) Jacob Male on ...

Python
~~Update ACQ Code~~
Completed by (s) Jacob Male on ...

Python
~~Implement Data Carving~~
Completed by (s) Jacob Male on ...

Sprint 13 (April 5th - 18th)

+ Add task

Hide completed 4

Python
~~Sleuthkit Implimentaion~~
Completed by (s) Jacob Male on ...

Python
~~Bug Fixing and Prjoect Structure~~
Completed by (s) Jacob Male on ...

Python
~~Update Data Craving With Photorec & Foremost~~
Completed by (s) Jacob Male on ...

Python
~~Impliment PDF File Analysis~~
Completed by (s) Jacob Male on ...

Sprint 14 (Apr 19th - 2nd)

+ Add task

Hide completed 4

~~Final Testing For Hashcat~~
Completed by (s) Jacob Male on ...

Python
~~Add Help Page~~
05/03 5/5
Completed by (s) Jacob Male on ...

Python
~~Impliment Hashcat (Password Cracking)~~
05/03 1/1
Completed by (s) Jacob Male on ...

Testing
~~Fix Bugs In Sluethkit Implimentation~~
Completed by (s) Jacob Male on ...

Sprint 15 (May 3rd - 16 (& 17))

+ Add task

Docs
~~Application Screenshots~~
Completed by (s) Jacob Male on ...

Docs
~~Installation & Usage Doc~~
Completed by (s) Jacob Male on ...

Docs
~~Create Usage Video (5 Mins)~~
Completed by (s) Jacob Male on ...

Docs
~~Duplicate This Planner To Github Planner~~
Completed by (s) Jacob Male on ...

Docs Testing
~~User Testing~~
4/4
Completed by (s) Jacob Male on ...

Docs
~~Uml Diagram~~
Completed by (s) Jacob Male on ...

Python

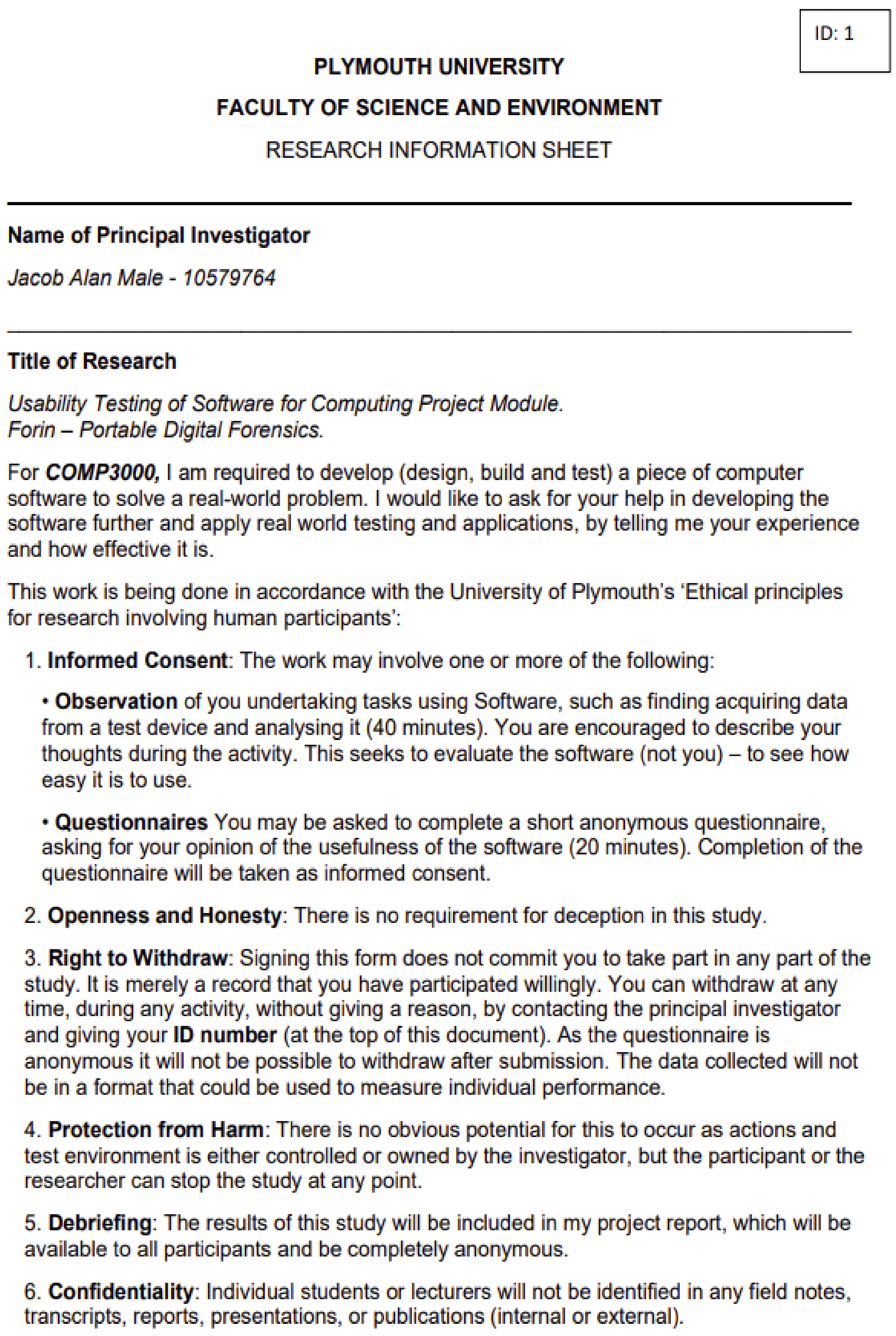
Backlog

+ Add task

Create "dummy PC" on USB Hard drive running windows 10
Day 1
Day 2
Day 3
Day 4
Day 5
Day 6
Day 7
11/15/2020 1/8

Appendix 2.4: *Sprint 11-15 (Late-Stage Development)*

Appendix 3: User Testing Documentation



ID: 1

PLYMOUTH UNIVERSITY

FACULTY OF SCIENCE AND ENVIRONMENT

RESEARCH INFORMATION SHEET

Name of Principal Investigator

Jacob Alan Male - 10579764

Title of Research

Usability Testing of Software for Computing Project Module.
Forin – Portable Digital Forensics.

For ***COMP3000,*** I am required to develop (design, build and test) a piece of computer software to solve a real-world problem. I would like to ask for your help in developing the software further and apply real world testing and applications, by telling me your experience and how effective it is.

This work is being done in accordance with the University of Plymouth's 'Ethical principles for research involving human participants':

1. **Informed Consent**: The work may involve one or more of the following:

   • **Observation** of you undertaking tasks using Software, such as finding acquiring data from a test device and analysing it (40 minutes). You are encouraged to describe your thoughts during the activity. This seeks to evaluate the software (not you) – to see how easy it is to use.

   • **Questionnaires** You may be asked to complete a short anonymous questionnaire, asking for your opinion of the usefulness of the software (20 minutes). Completion of the questionnaire will be taken as informed consent.

2. **Openness and Honesty**: There is no requirement for deception in this study.

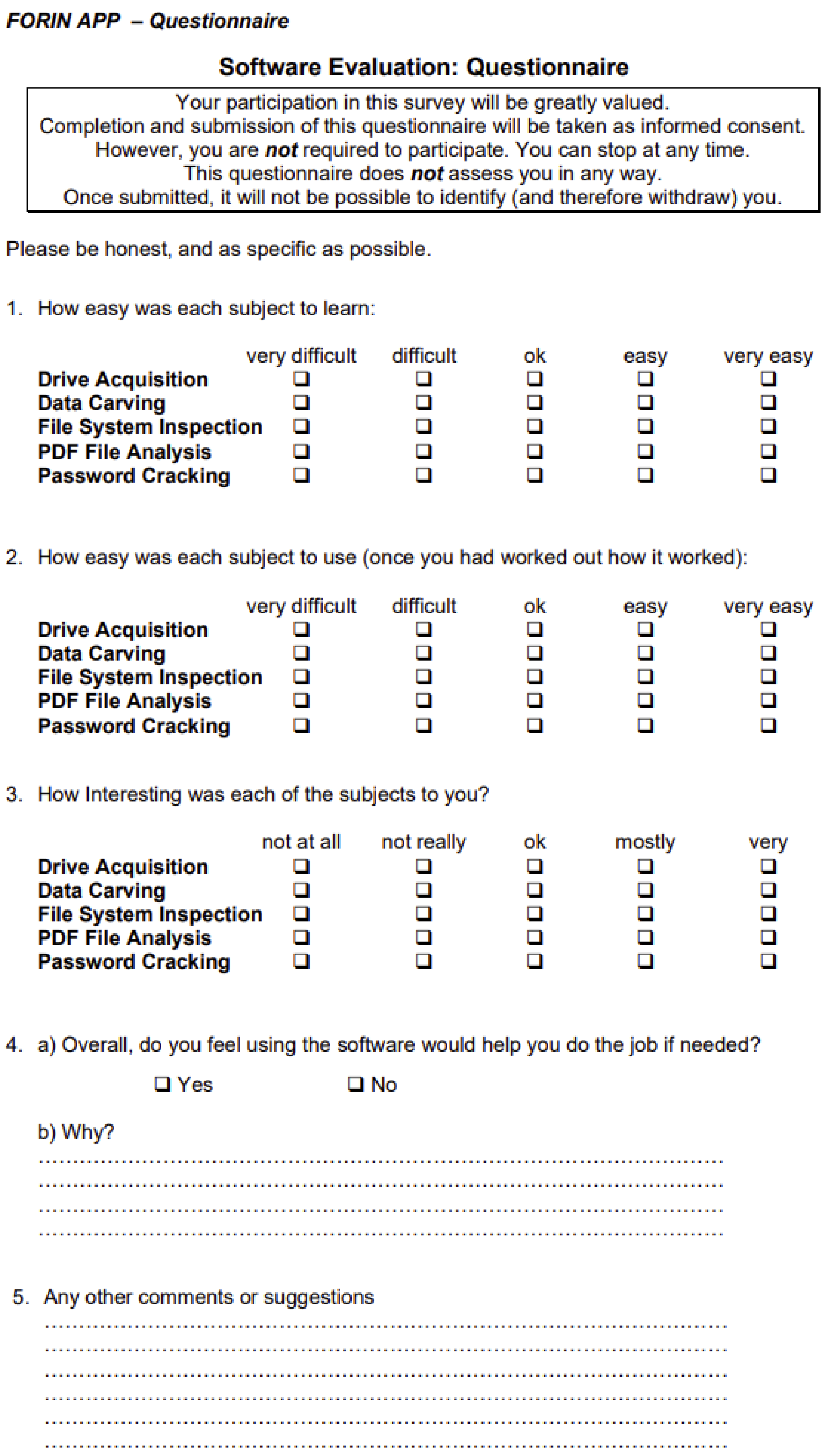
3. **Right to Withdraw**: Signing this form does not commit you to take part in any part of the study. It is merely a record that you have participated willingly. You can withdraw at any time, during any activity, without giving a reason, by contacting the principal investigator and giving your **ID number** (at the top of this document). As the questionnaire is anonymous it will not be possible to withdraw after submission. The data collected will not be in a format that could be used to measure individual performance.

4. **Protection from Harm**: There is no obvious potential for this to occur as actions and test environment is either controlled or owned by the investigator, but the participant or the researcher can stop the study at any point.

5. **Debriefing**: The results of this study will be included in my project report, which will be available to all participants and be completely anonymous.

6. **Confidentiality**: Individual students or lecturers will not be identified in any field notes, transcripts, reports, presentations, or publications (internal or external).

Appendix 3.1: *Forin Research Information Sheet*



FORIN APP – Questionnaire

Software Evaluation: Questionnaire

Your participation in this survey will be greatly valued.
Completion and submission of this questionnaire will be taken as informed consent.
However, you are ***not*** required to participate. You can stop at any time.
This questionnaire does ***not*** assess you in any way.
Once submitted, it will not be possible to identify (and therefore withdraw) you.

Please be honest, and as specific as possible.

1. How easy was each subject to learn:

	very difficult	difficult	ok	easy	very easy
Drive Acquisition	☐	☐	☐	☐	☐
Data Carving	☐	☐	☐	☐	☐
File System Inspection	☐	☐	☐	☐	☐
PDF File Analysis	☐	☐	☐	☐	☐
Password Cracking	☐	☐	☐	☐	☐

2. How easy was each subject to use (once you had worked out how it worked):

	very difficult	difficult	ok	easy	very easy
Drive Acquisition	☐	☐	☐	☐	☐
Data Carving	☐	☐	☐	☐	☐
File System Inspection	☐	☐	☐	☐	☐
PDF File Analysis	☐	☐	☐	☐	☐
Password Cracking	☐	☐	☐	☐	☐

3. How Interesting was each of the subjects to you?

	not at all	not really	ok	mostly	very
Drive Acquisition	☐	☐	☐	☐	☐
Data Carving	☐	☐	☐	☐	☐
File System Inspection	☐	☐	☐	☐	☐
PDF File Analysis	☐	☐	☐	☐	☐
Password Cracking	☐	☐	☐	☐	☐

4. a) Overall, do you feel using the software would help you do the job if needed?

   ☐ Yes ☐ No

   b) Why?

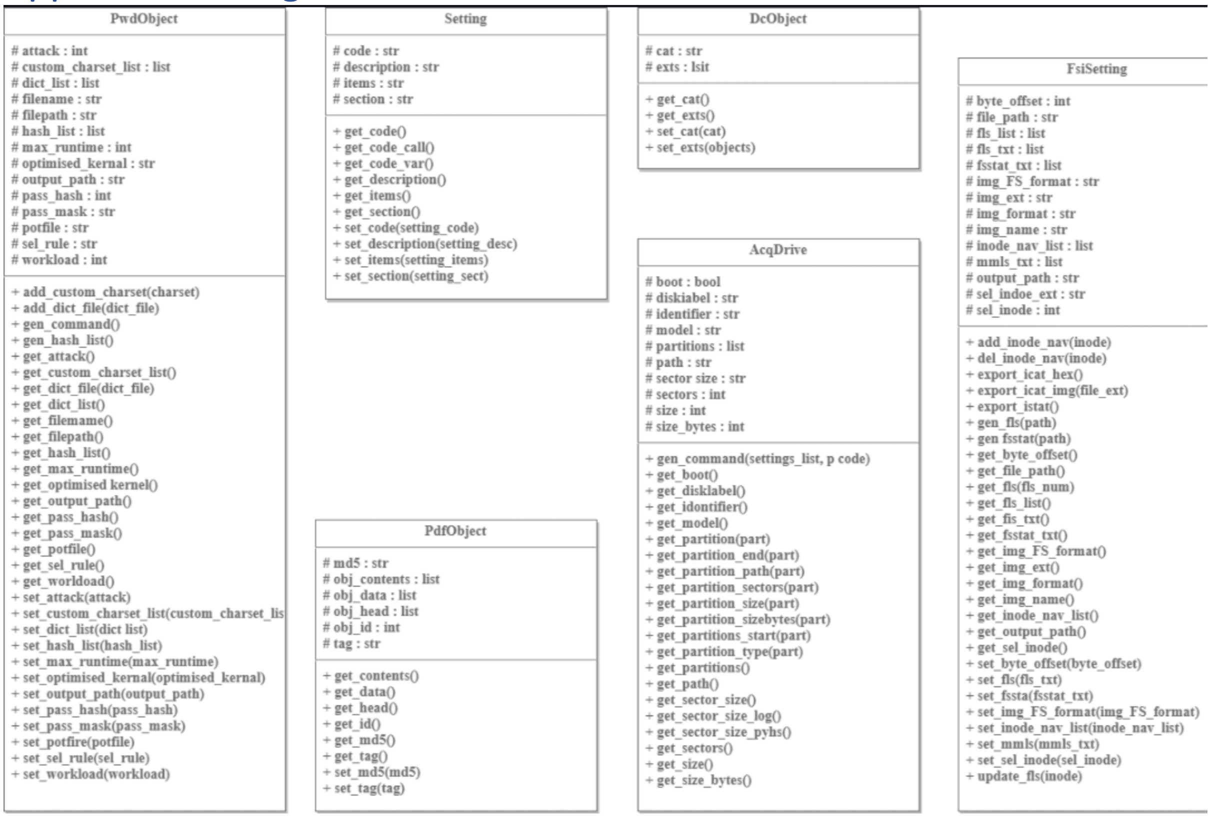
   ....
   ....
   ....
   ....

5. Any other comments or suggestions

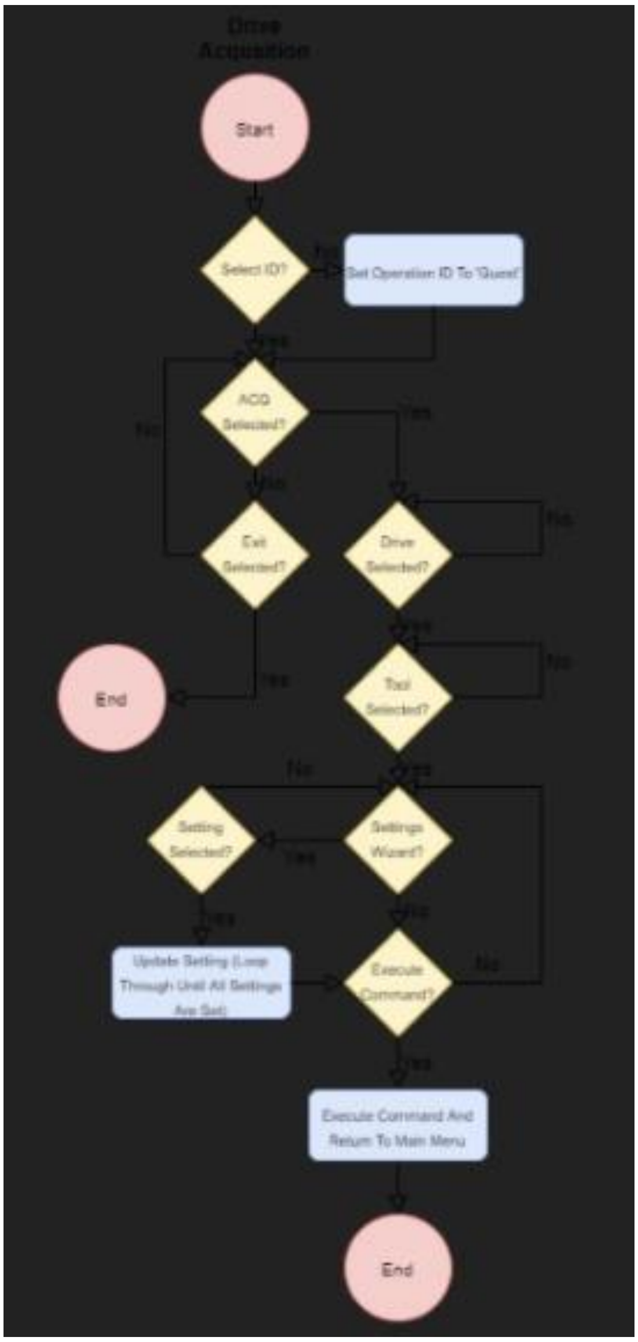
   ....
   ....
   ....
   ....
   ....
   ....

Appendix 3.2: *Forin Questionnaire Sheet Template*

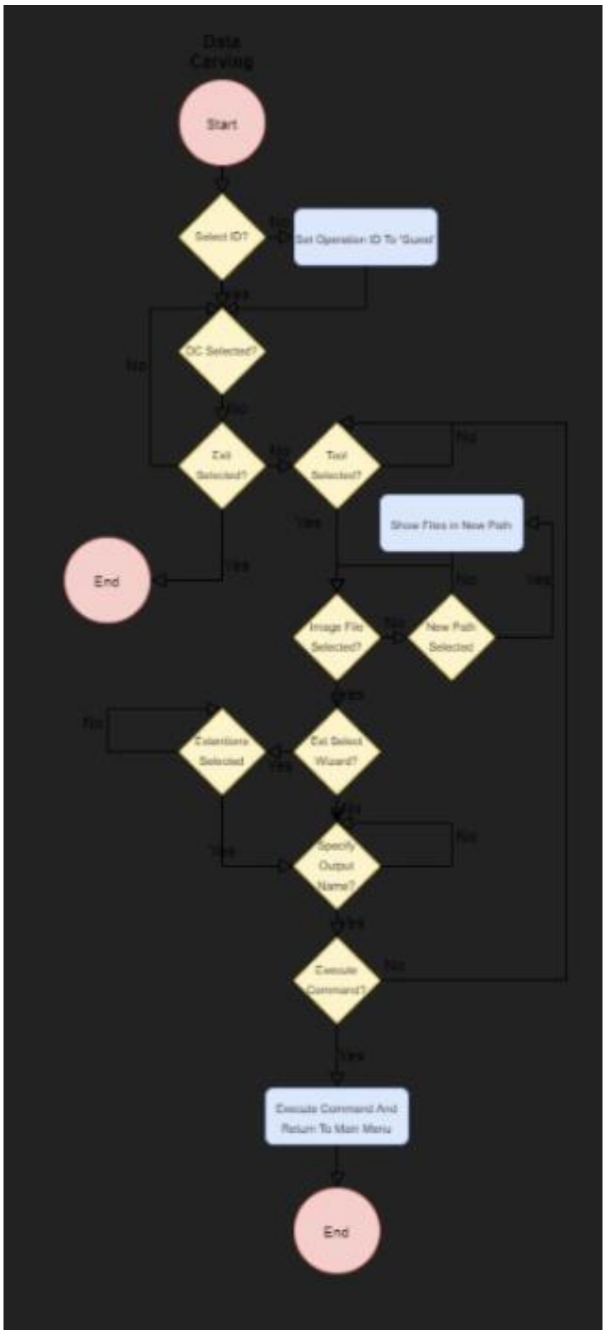
Appendix 4: Usage UMLs



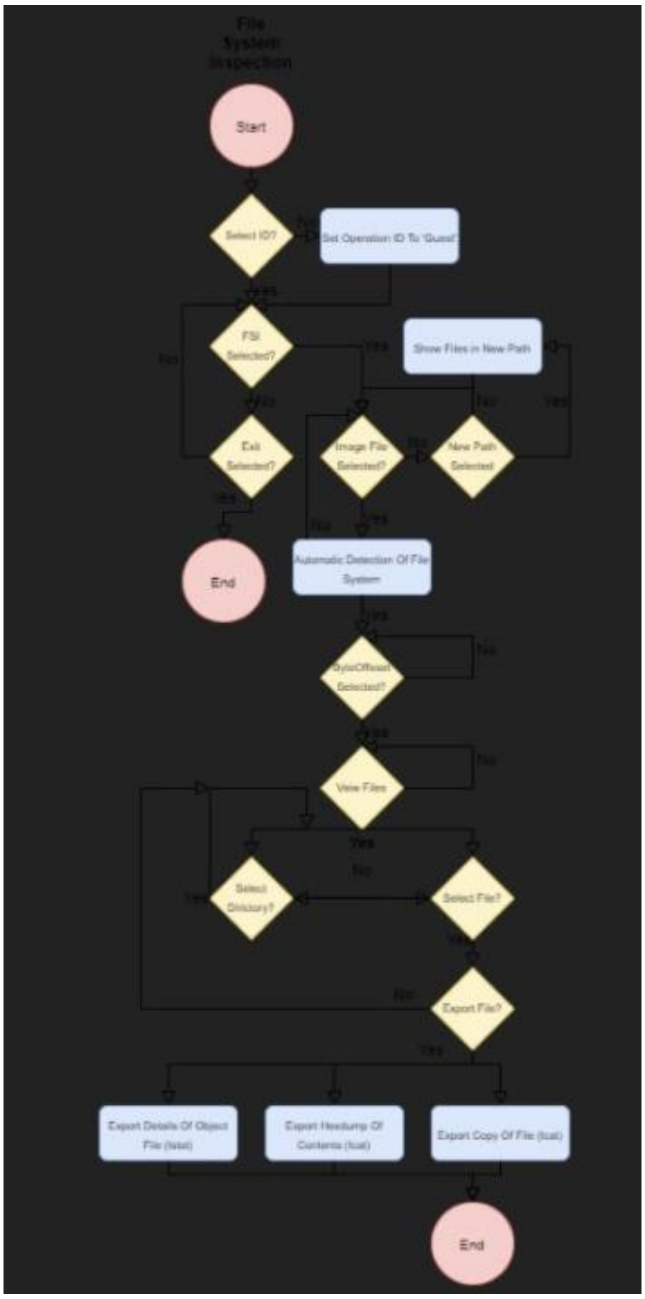
Appendix 4.1: *Forin Classes UML*



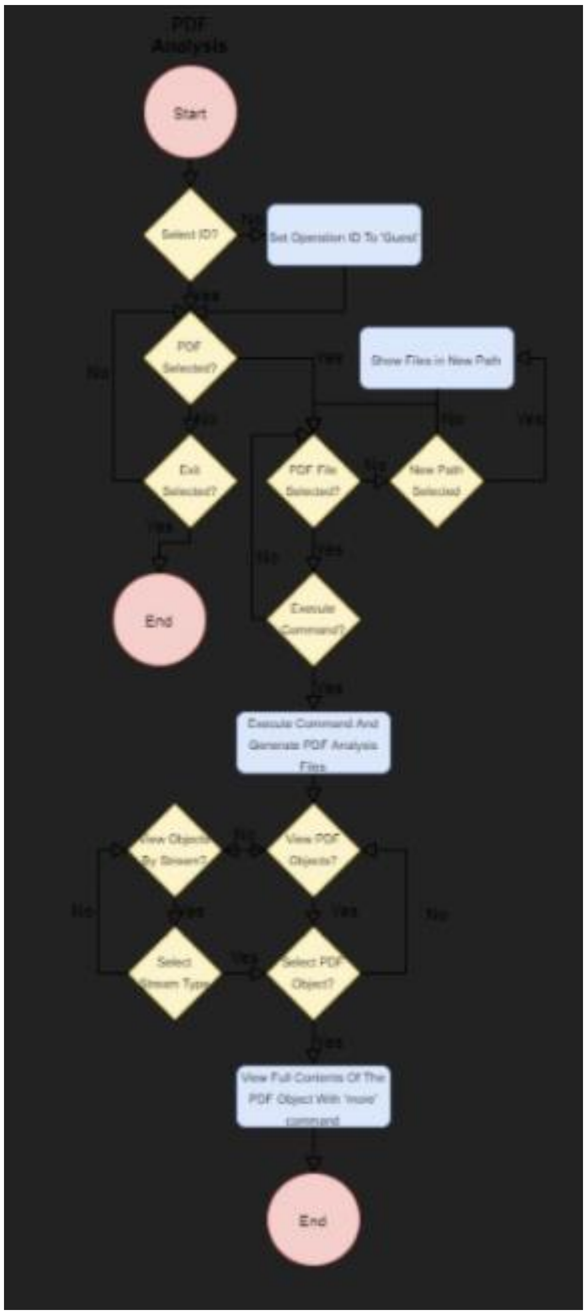
Appendix 4.2: *Drive Acquisition Usage Flowchart*



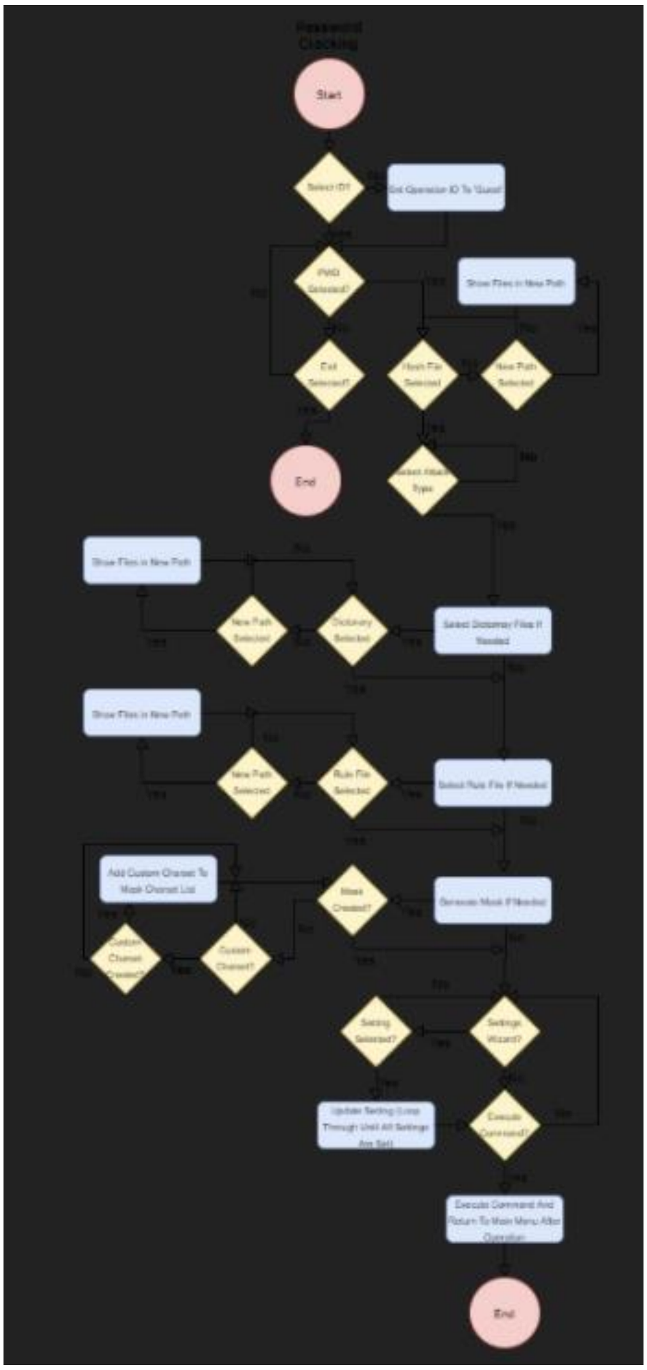
Appendix 4.3: *Data Carving Usage Flowchart*



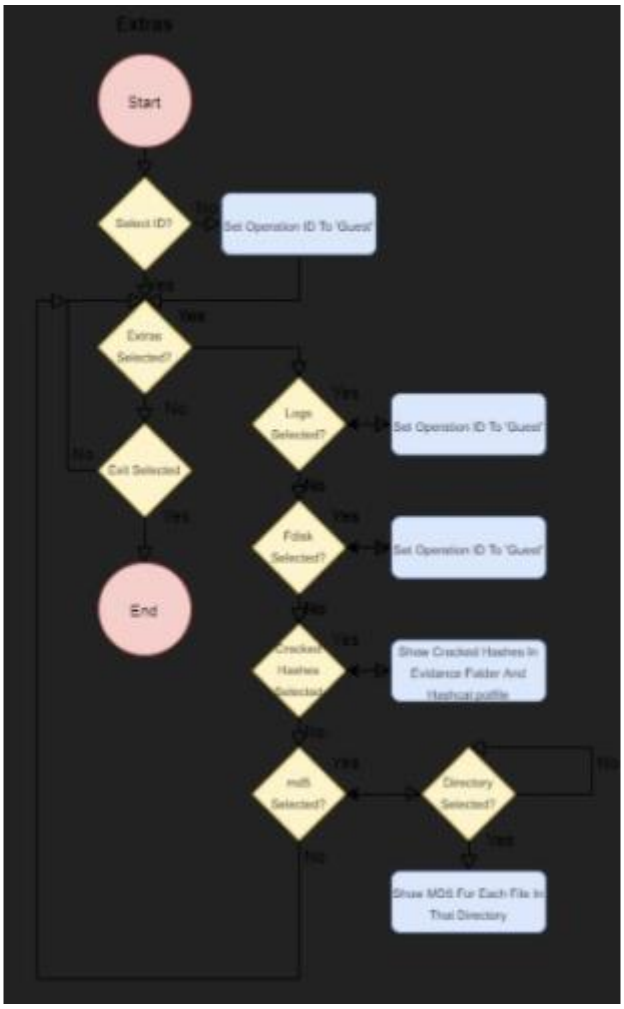
Appendix 4.4: *File System Inspection Usage Flowchart*



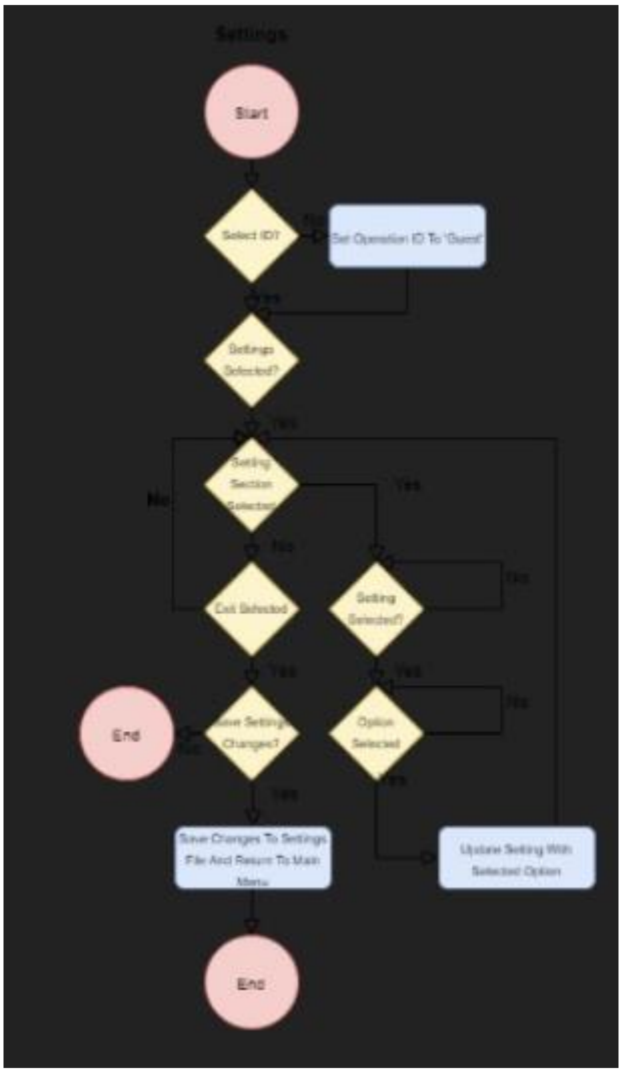
Appendix 4.5: *PDF file Analysis Usage Flowchart*



Appendix 4.6: *Hash Cracking Usage Flowchart*

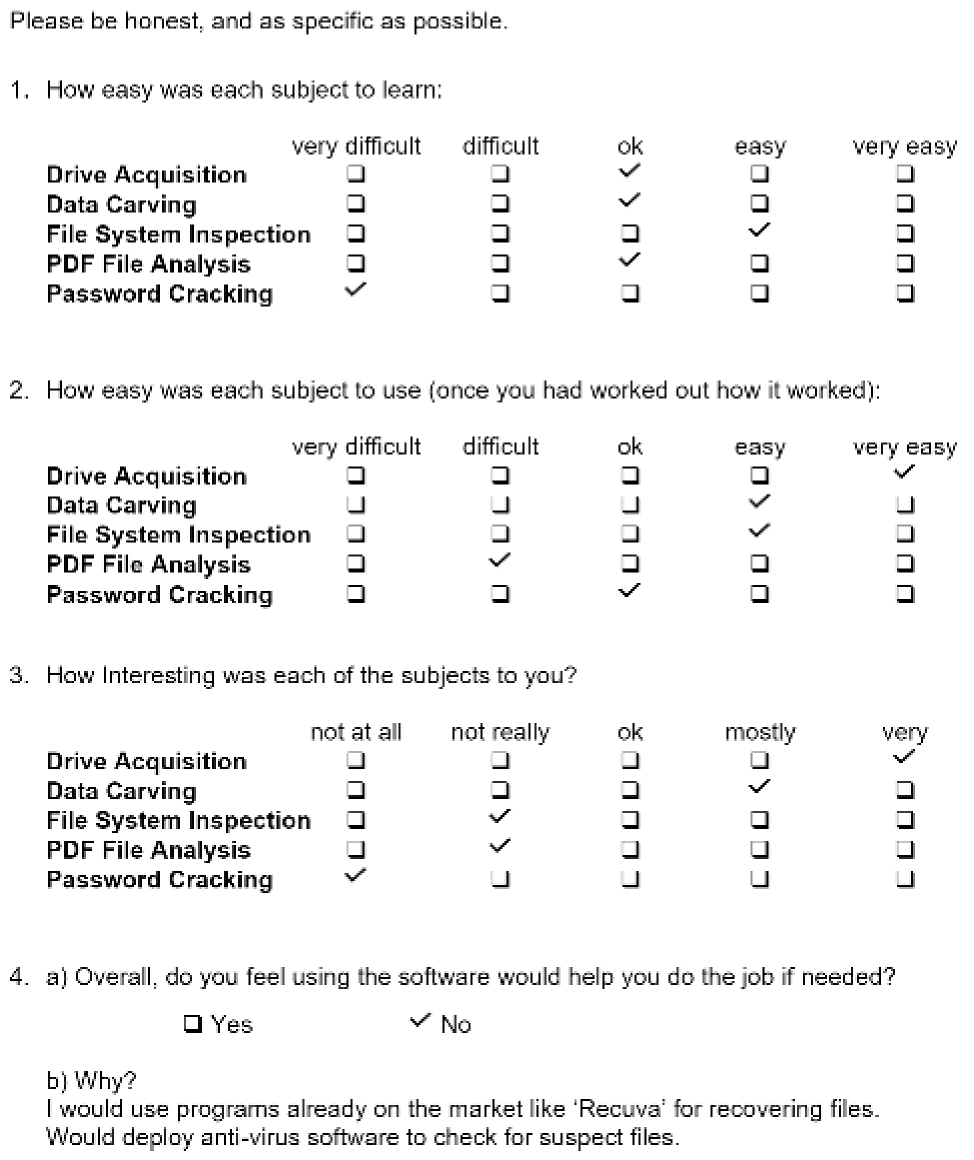

Extras
Start
Select ID?
No
Set Operation ID To 'Guest'
Yes
Extras Selected?
Yes
No
Exit Selected
No
Yes
End
Logs Selected?
Yes
Set Operation ID To 'Guest'
No
Fdisk Selected?
Yes
Set Operation ID To 'Guest'
No
Cracked Hashes Selected
Yes
Show Cracked Hashes In Evidence Folder And Hashcat potfile
No
md5 Selected?
Yes
Directory Selected?
No
Yes
Show MD5 For Each File In That Directory
No

Appendix 4.7: *Extras Usage Flowchart*



Appendix 4.8: *Settings Usage Flowchart*

Appendix 5: User Testing Results



Please be honest, and as specific as possible.

1. How easy was each subject to learn:

	very difficult	difficult	ok	easy	very easy
Drive Acquisition	❑	❑	✓	❑	❑
Data Carving	❑	❑	✓	❑	❑
File System Inspection	❑	❑	❑	✓	❑
PDF File Analysis	❑	❑	✓	❑	❑
Password Cracking	✓	❑	❑	❑	❑

2. How easy was each subject to use (once you had worked out how it worked):

	very difficult	difficult	ok	easy	very easy
Drive Acquisition	❑	❑	❑	❑	✓
Data Carving	❑	❑	❑	✓	❑
File System Inspection	❑	❑	❑	✓	❑
PDF File Analysis	❑	✓	❑	❑	❑
Password Cracking	❑	❑	✓	❑	❑

3. How Interesting was each of the subjects to you?

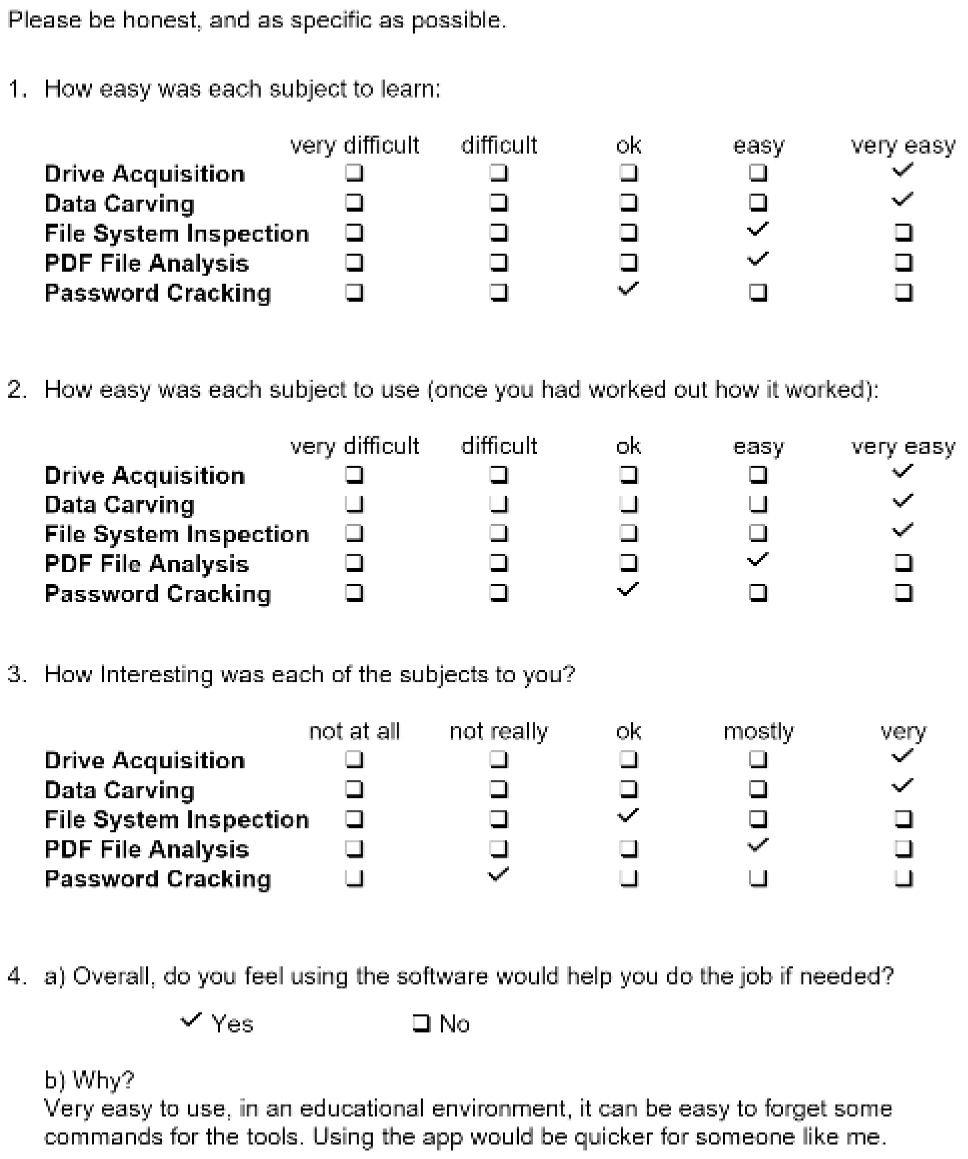
	not at all	not really	ok	mostly	very
Drive Acquisition	❑	❑	❑	❑	✓
Data Carving	❑	❑	❑	✓	❑
File System Inspection	❑	✓	❑	❑	❑
PDF File Analysis	❑	✓	❑	❑	❑
Password Cracking	✓	❑	❑	❑	❑

4. a) Overall, do you feel using the software would help you do the job if needed?

   ❑ Yes [✓ No]

   b) Why?
   I would use programs already on the market like 'Recuva' for recovering files.
   Would deploy anti-virus software to check for suspect files.

Appendix 5.1: *Questionnaire results 1*



Please be honest, and as specific as possible.

1. How easy was each subject to learn:

	very difficult	difficult	ok	easy	very easy
Drive Acquisition	❑	❑	❑	❑	✓
Data Carving	❑	❑	❑	❑	✓
File System Inspection	❑	❑	❑	✓	❑
PDF File Analysis	❑	❑	❑	✓	❑
Password Cracking	❑	❑	✓	❑	❑

2. How easy was each subject to use (once you had worked out how it worked):

	very difficult	difficult	ok	easy	very easy
Drive Acquisition	❑	❑	❑	❑	✓
Data Carving	❑	❑	❑	❑	✓
File System Inspection	❑	❑	❑	❑	✓
PDF File Analysis	❑	❑	❑	✓	❑
Password Cracking	❑	❑	✓	❑	❑

3. How Interesting was each of the subjects to you?

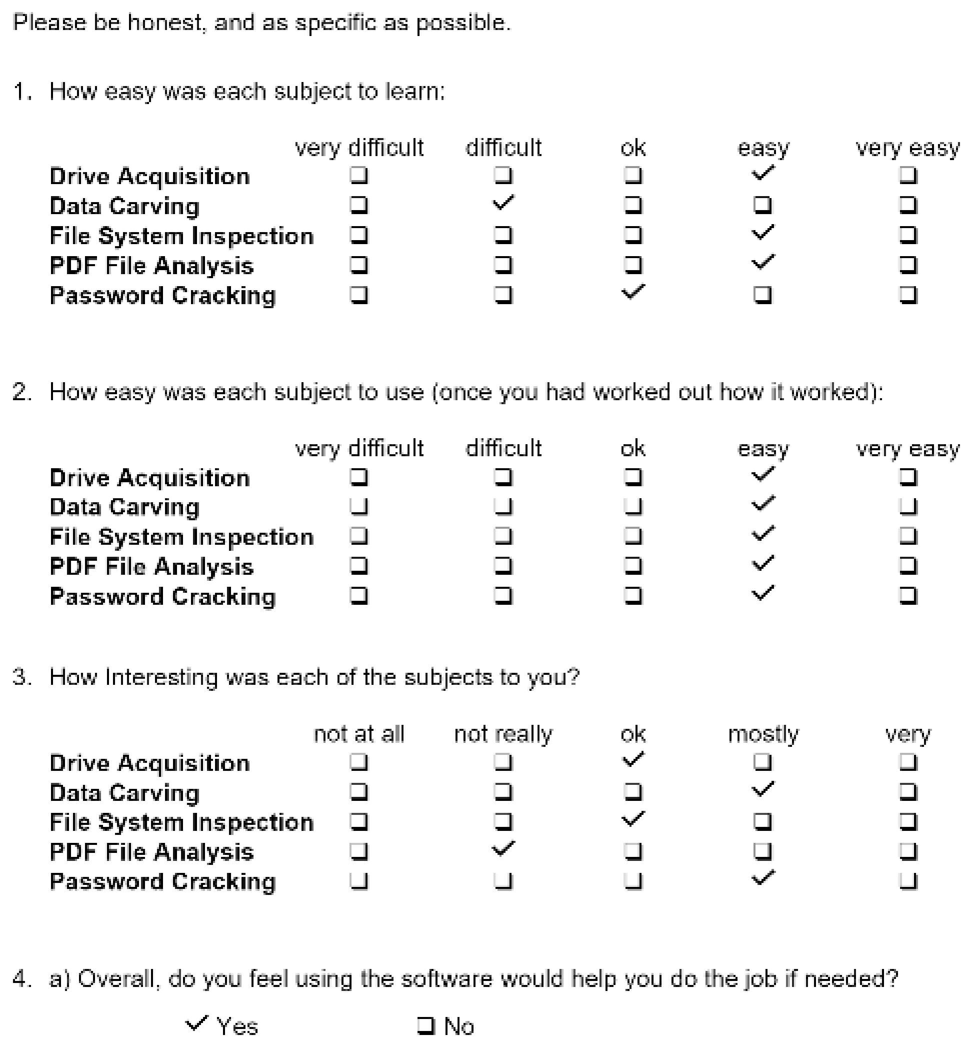
	not at all	not really	ok	mostly	very
Drive Acquisition	❑	❑	❑	❑	✓
Data Carving	❑	❑	❑	❑	✓
File System Inspection	❑	❑	✓	❑	❑
PDF File Analysis	❑	❑	❑	✓	❑
Password Cracking	❑	✓	❑	❑	❑

4. a) Overall, do you feel using the software would help you do the job if needed?

   [✓ Yes] ❑ No

   b) Why?
   Very easy to use, in an educational environment, it can be easy to forget some commands for the tools. Using the app would be quicker for someone like me.

Appendix 5.2: *Questionnaire results 2*



Please be honest, and as specific as possible.

1. How easy was each subject to learn:

	very difficult	difficult	ok	easy	very easy
Drive Acquisition	❑	❑	❑	✓	❑
Data Carving	❑	✓	❑	❑	❑
File System Inspection	❑	❑	❑	✓	❑
PDF File Analysis	❑	❑	❑	✓	❑
Password Cracking	❑	❑	✓	❑	❑

2. How easy was each subject to use (once you had worked out how it worked):

	very difficult	difficult	ok	easy	very easy
Drive Acquisition	❑	❑	❑	✓	❑
Data Carving	❑	❑	❑	✓	❑
File System Inspection	❑	❑	❑	✓	❑
PDF File Analysis	❑	❑	❑	✓	❑
Password Cracking	❑	❑	❑	✓	❑

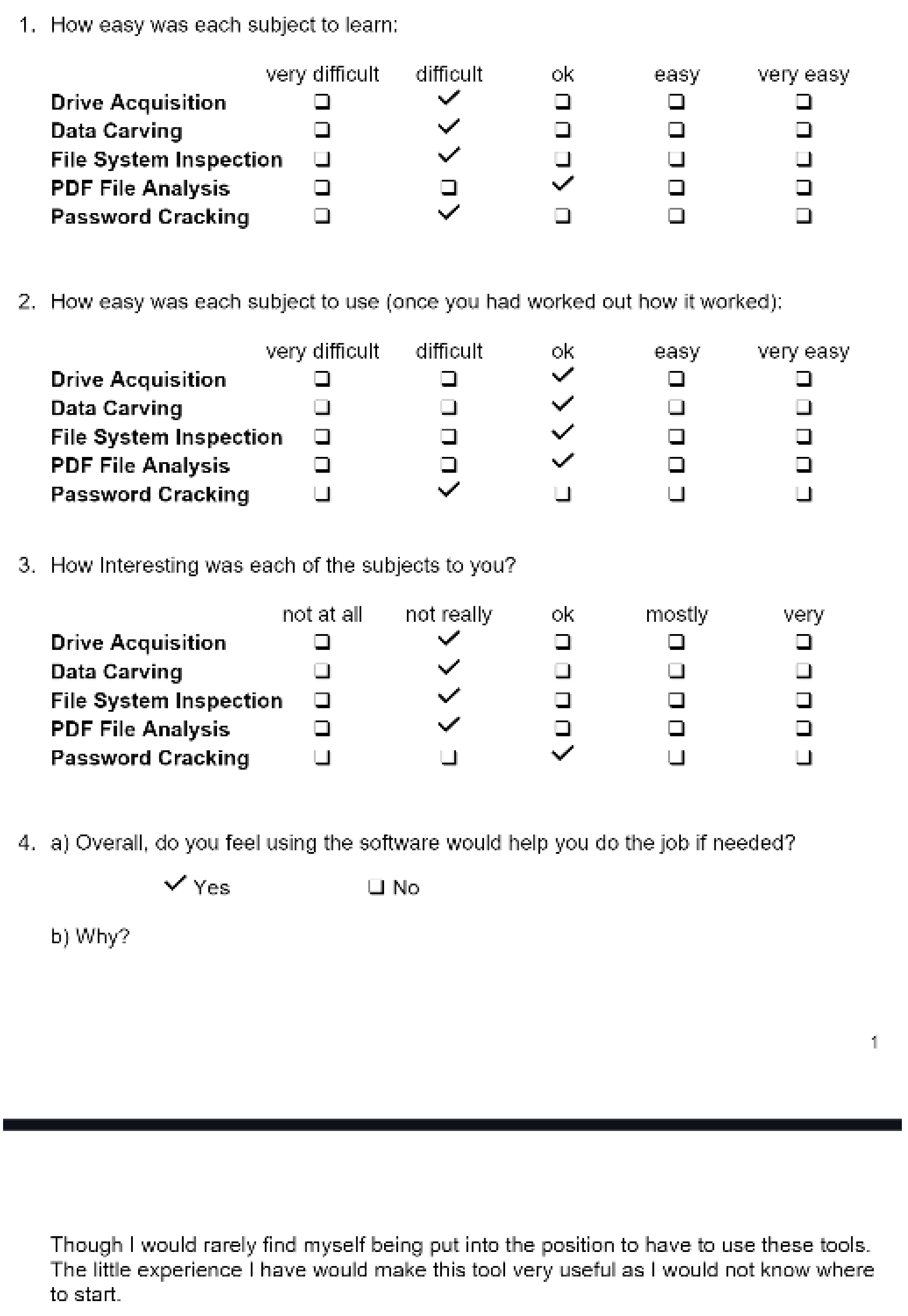
3. How Interesting was each of the subjects to you?

	not at all	not really	ok	mostly	very
Drive Acquisition	❑	❑	✓	❑	❑
Data Carving	❑	❑	❑	✓	❑
File System Inspection	❑	❑	✓	❑	❑
PDF File Analysis	❑	✓	❑	❑	❑
Password Cracking	❑	❑	❑	✓	❑

4. a) Overall, do you feel using the software would help you do the job if needed?

   [✓ Yes] ❑ No

Appendix 5.3: *Questionnaire results 3*



1. How easy was each subject to learn:

	very difficult	difficult	ok	easy	very easy
Drive Acquisition	❑	✓	❑	❑	❑
Data Carving	❑	✓	❑	❑	❑
File System Inspection	❑	✓	❑	❑	❑
PDF File Analysis	❑	❑	✓	❑	❑
Password Cracking	❑	✓	❑	❑	❑

2. How easy was each subject to use (once you had worked out how it worked):

	very difficult	difficult	ok	easy	very easy
Drive Acquisition	❑	❑	✓	❑	❑
Data Carving	❑	❑	✓	❑	❑
File System Inspection	❑	❑	✓	❑	❑
PDF File Analysis	❑	❑	✓	❑	❑
Password Cracking	❑	✓	❑	❑	❑

3. How Interesting was each of the subjects to you?

	not at all	not really	ok	mostly	very
Drive Acquisition	❑	✓	❑	❑	❑
Data Carving	❑	✓	❑	❑	❑
File System Inspection	❑	✓	❑	❑	❑
PDF File Analysis	❑	✓	❑	❑	❑
Password Cracking	❑	❑	✓	❑	❑

4. a) Overall, do you feel using the software would help you do the job if needed?

[✓ Yes]   ❑ No

b) Why?

1

Though I would rarely find myself being put into the position to have to use these tools. The little experience I have would make this tool very useful as I would not know where to start.

Appendix 5.4: *Questionnaire results 4*