Automated Eforensics Analysis System

SWE40001
SOFTWARE ENGINEERING PROJECT A
SEMESTER 1, 2023
Project Plan

Name	Position	Email
Mafaz Abrar Jan Chowdhury	Team Leader	103172407@student.swin.edu.au
Harrison Zito	Documentation Manager	103591785@student.swin.edu.au
Joe Sutton Preece	Repository Champion	102568393@student.swin.edu.au
Thanh Nguyen	Sprint Master	101607169@student.swin.edu.au
Robert Findlay	Technical Lead	103613414@student.swin.edu.au
S M Ragib Rezwan	End to End Champion	103172423@student.swin.edu.au

Document Change Control

Version	**Date**	**Authors**	**Summary of Changes**
1.00	08/03/23	All Authors	Initial draft of draft created.
1.10	20/03/23	All Authors	Updating draft based on design changes.
1.20	26/03/23	All Authors	Updating draft based on Supervisor feedback.

Document Sign Off

Name	**Position**	**Signature**	**Date**
Irene Moser	Project Supervisor		
Mafaz Abrar Jan Chowdhury	Team Leader	Mafaz	26/03/23
Harrison Zito	Documentation Manager	hz	26/03/23
Joe Sutton Preece	Repository Champion	JRSP	26/03/23
Thanh Nguyen	Sprint Master	TN	26/03/23
Robert Findlay	Technical Lead	RF	26/03/23
S M Ragib Rezwan	End to End Champion	Ragib	26/03/23

Client Sign Off

Name	**Position**	**Signature**	**Date**
Allan Ou	Co-Founder		
Organisation. Bad Security Inc			

Name	**Position**	**Signature**	**Date**
Tom Maloney	Co-Founder		
Organisation. Bad Security Inc			

Contents

Chapter 1

INTRODUCTION

The project plan is a formal document that presents information on the control and execution of the project "Automated e-Forensic analysis system." The client, development team, stakeholder and supervisor of the development team should read this project plan. Readers of this project plan will be able to understand development methodology, product delivery, project expectations, risk management and stakeholder product.

Project context is first established, identifying the key stakeholders and team roles. The scope is defined to ensure that client and team expectations of deliverables are mutually agreed upon. Following this, the critical success factors and acceptance criteria communicate the goals and objectives of the project.

The Project processes and procedures are established along with organisational structures to ensure cohesive team function and that best coding and operating practices are followed.

To allow for the successful completion of the project an initial project schedule is established, and a brief assessment of the key risks and mitigation strategies is conducted along with an estimated budget.

1.1 Background

"Automated e-Forensic analysis system" is a Capstone project introduced by Allan O. and his work partner Tom Maloney with the main inspiration coming from Allan's university student life. During Allan's final year of university, there were barely any cybersecurity-related projects to be involved. Therefore, he was inspired to create Bad Security Inc. to produce more projects for Swinburne cybersecurity students.

Furthermore, Allan O. wanted the "Automated e-Forensic analysis system" project to produce an e-forensic drive analysis software. Hence, Allan's vision is to combine various analysis and mounting drive tools into a user-friendly application to be used by e-Forensics users.

1.2 Key Project Personnel

The key personnel involved in this project are as follows:

1.2.1 Client

Allan O. of Bad Security Inc. is a graduated Bachelor of Computer Science (Cybersecurity) Swinburne student who is currently working as a Cyber Security Cloud Engineer at Deloitte.

Tom Maloney of Bad Security Inc. is a graduated Bachelor of Computer Science (Professional) (Cybersecurity) Swinburne student who is a Security Engineer at Triskele Labs.

1.2.2 Other Stake holders

E-Forensics Investigator: This stakeholder is an expert working professionally in the e-Forensic field. They want to utilize the "Automated e-Forensic analysis system" software project for their business needs.

Customer with e-Forensic background: This stakeholder will have different levels of knowledge within the e-Forensic field. They want to use the "Automated e-Forensic analysis system" software project for educational or personal purposes.

1.2.3 Project Supervisor, Team Leader, and Key Project Members

Name	**Role**
Irene Moser	Project Supervisor
Mafaz Abrar Jan Chowdhury	Team Leader
Harrison Zito	Documentation Manager
Joe Sutton Preece	Repository Champion
Thanh Nguyen	Sprint Master
Robert Findlay	Technical Lead
S M Ragib Rezwan	End to End Champion

Table 1.1: Roles

Chapter 2

TERMS OF REFERENCE

The project aims to automate a set of forensic tasks. The client envisions that a skilled forensic analyst should be able to utilize the tools to improve the efficiency of forensic tasks.

It is the responsibility of the team to deliver an application that can identify suspicious files on a forensic format drive. Suspicious files are those that have been marked for deletion, renamed or are unallocated in the drive as outlined in 2.2.1.

A completed application should be delivered to the client at the conclusion of the project outlined in Section 7 Schedule.

2.1 Objectives

The objectives of this project:

1. Build a CLI application that automates the collection of suspicious files as defined in 2.2.1.
2. Generate timeline report and list of suspicious files (PDF or CSV)
3. Build Interactive GUI application.
4. Perform MD5 and SHA1 hashing and validation.
5. Automate mounting of forensic drive.

2.2 Scope

Earliest Start: 06-03-2023

Latest Finish: 29-10-2023

The scope of the project will include the production of software that will automate file collection of a drive, and produce reports based on the collection. Two primary reports will be produced: A list of suspicious files, and a timeline of file MACs. The file collection will be conducted on a variety of drive types as listed below.

2.2.1 Within Scope

The In-Scope objects will mainly include all items necessary to create a GUI application (with an integrated CLI) that automates the processes that are critical for performing forensics on a drive. This would include: mounting the files (in E01, DD, LEF, ZIP, and DMG formats); creating MD5 and SHA1 hashes of the drives; identification of any suspicious files (any file that has been modified or deleted without the user or system knowledge) and functionality of carving out files, searching files with the Content Header (PK) or any specific keyword, listing suspicious files' details (names, iNode number, MAC dates, File Size and MD5 hashes). Furthermore, it will also include the generation of a report (in PDF and CSV format) that would include: a list of files that were manipulated, a Timeline of the file manipulation, the manipulation method and also the time zone of the user and the computer

2.2.2 Without Scope

The out-of-scope objects will include everything that is outside the boundaries of the project criteria and requirements. This will include further analysis and processing of reports, identification of relevant files to a particular case, live analysis of a system, collection of non-file information such as events, efficient processing of large drives, and batch processing of multiple drives.

2.2.3 Tentatively in Scope

The tentatively in Scope objects mainly consist of the optional requirements suggested to the team by the client. These are not necessary for the project to be deemed complete but more like "moon shot" type goals. These include an interactive report (reports within applications in addition to PDF, and CSV), the creation of forensic files from the physical drives, memory forensic collections (forensic format from memory dump, identification of suspicious activity), and email content scans based on attachments and keywords.

2.3 Critical Success Factors

Critical Success Factors are the factors or objectives that must be completed at minimum to ensure the project can be deemed complete

1. The software is accompanied by a UI that grants access to the complete set of features as the CLI version.
2. The performance of the software must be such that it can analyse the forensic formats outlined in 2.2.1 The software must be able to categorise suspicious files according to deleted, renamed, and carved as outlined in 2.2.1
3. No changes made to drive after software collection, thus validation of MD5 produced hash from drive must pass in 100% of test cases. The software must produce PDF reports and CSV reports based on the categorised files as identified in 2.2.1.

2.4 Acceptance Criteria

The client requires that the software can automate the manual tasks of file collection, thus the functions present must automate the tasks outlined in section 2.2.1.

The client will be able to determine the acceptability of the software based on the objectives outlined in section 2.1 If the functionality achieves the functional requirements of those objectives, the software should be accepted.

Acceptance testing will be conducted through the collection of user stories based on the objectives and scope. Testing will be conducted both internally and externally with feedback and changes made at each phase. The outcome of the tests should meet the requirements of the critical success factors, then the acceptance tests will be passed.

Chapter 3

ESTABLISHMENT

3.1 Processes, Procedures and Standards

3.1.1 Software Development Methodology

The project development methodology will be based on a modified version of Scrum to facilitate the required schedule for deliverables. The project will be split into phases where the beginning phases consist of planning and the later phases will consist of development which will be conducted in sprints.

Scrum

This project will use the Scrum project management framework, based on AGILE project development principles.

Scrum was chosen due to its success in previous large scale development projects.

Due to the team's general lack of experience in large scale enterprise application development, it can be assumed that, over the course of development, there will be a disproportionately large number of unexpected hurdles and obstacles, both in terms of the skill level of the team with the chosen toolsets, as well as complexities with the toolsets themselves, classified as Major Skill Roadblocks and Minor Skill Roadblocks in section 6.1.

Scrum prioritizes incremental development with frequent releases. This makes the process especially resilient to the form of roadblocks most likely to be encountered. At the end of each stage of development, the team can discuss obstacles and decide on appropriate solutions, while still delivering functionality regularly. Quality Assurance and Testing The requirements for this project include several acceptance criteria outlined in section 2.4.

Once a Sprint Backlog item has completed development, the item should be marked for testing. The responsibility for testing the source code falls to the developer writing the source code, however, the tests must be based on the acceptance criteria to a reasonable extent. The developer must include in docstrings/comments how each acceptance criterion was addressed, and how the feature passes/fails each criterion.

The testing library used should be the standard testing library for the language chosen, to be agreed upon by the team before development. The developer should write their own unit tests and integration tests using this library. The directory structure must consider testing files as well during the initial infrastructure setup.

3.1.2 Versioning Systems

The file versioning systems used will be GitHub and OneDrive.

GitHub Standards

The team will use GitHub for versioning of the project source code, as well as for the maintenance of published administrative documents. A single, private repository should be created and maintained before project development begins. Each member should be invited to the repository as a maintainer.

Team members should use GitHub accounts associated with their student emails (not personal emails) to contribute to the repository. On completion of the project, the repository must be returned to the client through a transfer of ownership.

OneDrive Standards

The team will use OneDrive to store meeting minutes, meeting agendas, team work logs, and any team resources. The OneDrive should also be used to store working drafts for administrative and management documents before they are published to Github. The supervisor, as well as all team members, should be allowed access to the OneDrive directory.

3.1.3 GUI Design Process

One of the application requirements is a GUI Interface. To effectively design a GUI Interface, developers need clear lines of communication with clients and end users. However, end-user focused design for the GUI Interface is an optional requirement for this project. Therefore, the project will not be using a structured User Centred Design process for the GUI design.

Instead, the design of the GUI will be integrated into the Sprint-based development framework. The GUI will be split into actionable features, rather than an overarching initial design. Each feature will be designed, developed and tested following Scrum principles.

Due to the liberty provided by the clients in customizing the GUI, the design process will not conduct detailed end user surveys or interviews – the client review process for the designs will be conducted during the Sprint Review and Retrospective meetings for each Sprint, where client feedback will be integrated into the Product Backlog.

3.1.4 Coding Standards

The chosen language for the project will inform the coding standards followed. This will determine the naming structure for variables, functions, and objects.

Comments: code should be commented on to ensure mutual understanding of functionality. Documentation comments should be included with classes, and functions. And inline comments should be included where necessary.

The code should be written with collaboration and readability in mind. This means that variable, function and object names should favour descriptive names. Acronymized or shortened names that have unclear meanings are prohibited. This also applies to issues like magic numbers where the inputs to a function are unclear, instead, they should be defined elsewhere using global or variable names and parsed into the function.

3.2 Project Environment

3.2.1 Development Environments

The development environment for each team will be their respective home workstations. They are free to use their own code editors with custom settings, e.g. customized setups of VSCode, Atom, Sublime etc.

It should be noted, no team member should use an IDE for development (e.g. PyCharm). The reason is that IDEs tend to create project files automatically, and these will conflict with the existing infrastructure on GitHub.

Debugging Environments

The debugging environment will be a docker container. The docker files and docker compose files will be hosted on the GitHub repository. The team members will build the image locally using the docker files.

An initial version of the docker image must be created before the first sprint commences and may be amended as development continues to adapt to evolving development environments.

The docker image will be based on Debian 11. This distribution was chosen as it is representative of the operating system that the software will be run on in the production environment. The container will be configured to forward all graphical applications using WSL. This allows team members using Microsoft Windows to access the graphical user interface (GUI).

The image will also contain the required libraries and tools for development.

Docker allows a consistent development environment to be distributed to team members, as well as having (after initial configuration) no dependence on the team members' chosen operating systems.

Workplaces

The workplace for this project will be a hybrid model. Workplaces will include:

- Team members' homes
- University library rooms

Computers

Each team member will use their personal PC for development.

Accounts

- Each team member will need their Docker account.
- Each team member will need their own GitHub account.
- Each team member will need their own Discord account.

Deployment Environments

The deployment environment will be the industry standard deployment toolset that is most popularly used for the selected development environment, target programming language, and target deployment device.

3.3 Project Team Skill Development Requirements

To develop the application requirements, each team member must be well versed in the following tools:

1. The Linux command line interface, including common command line tools covered in the unit eForensic Fundamentals.
2. The sleuthkit command line utility:
   https://github.com/sleuthkit
   http://sleuthkit.org/sleuthkit/docs/api-docs/4.12.0//
3. Docker :
   https://sourceforge.net/projects/vcxsrv/
   https://docs.docker.com/desktop/install/windows-install/

Chapter 4

DELIVERABLES, ACTIVITIES AND CAPITAL RESOURCES

4.1 Deliverables

1. User Manual

   Full manual submitted upon the completion of the system's development and partial user manual submitted upon the completion of each of the system's functionality during each sprint

2. Source Code

   Source code will be submitted upon the completion of the system's development

3. Prototype

   Prototype will be submitted before the Live testing with the client's VM (pre-infected with RAT)

4. Test Documents

   Full test documents submitted upon the completion of the system's development and partial test document submitted upon the completion of each of the system's functionality during each sprint

5. Binaries

   Binaries will be submitted upon the completion of the system's development

6. Presentation Video

   Presentation video will be submitted upon the completion of the system's design and before the live testing of the software

7. Design Documentation

   An Unofficial Design document will be created before beginning the system's development and will be updated accordingly in each sprint. Once the system has been developed, the final version will be submitted as the official Design document

8. Software Requirements Specification

   Software requirements Specifications will be submitted before beginning the development of the system and will be updated accordingly in each sprint

9. Software Quality Assurance Plan

   Software Quality Assurance Plan will be submitted before beginning the development of the system

10. Project Plan

    This document is the Project Plan and is submitted as one of the first documents.

With the new eForensics software, user manuals for operations and troubleshooting will be provided containing all the necessary steps for each scenario and technical details for troubleshooting purposes. Also, before the system is tested in a live environment (using the Client's VM pre-infected with RAT), a brief training session/demonstration will be performed with the client, which would cover the following functionalities of the system:

- Mounting common eforensic file formats (E01, DD, LEF (l01), ZIP, and DMG)
- Creating MD5 hash of the drive
- Checking and verifying MD5 and SHA1 hashes of files
- Finding all files that have been deleted or modified in any way without the user's and system's knowledge
- Identifying file content headers (zip files "PK" header, etc.)
- Scanning for keywords within files relevant to the investigation
- Generating a report with the findings which has:

  An event timeline of the findings (last modification time, how change occurred, retracing file manipulation location)

  A list of all suspicious files, their MAC dates and details (File Name, iNode Number, Modified Date, Accessed Date, Created Date, File Size, MD5 hash of the files, etc.)

  The time zone of the user and the current computer time

In the case where all the above functionalities have been developed up (to the standards desired by the Client) with ample time to spare, the following optional functionalities are also being considered to be added to the deliverables List and thus the Sprints:

- Creation of an eforensics file from a physical drive (like USB)
- Performing Memory forensics (like creation of eforensic file for memory, dumping of current memory into eforensic file, checking for suspicious activities in memory, etc.)
- Scanning email content for any attachments (or even keywords) that are suspicious

4.2 Activities

In order to develop this system, the team has decided to use the Scrum model as it not only provides more transparency in the product being developed but also ensures that each individual functionality has been fully developed before moving to the next one. This has led to the development of the following phases:

Semester 1 Phases The following phases take place in semester 1.

Phase 1: Planning **Goal:** Gathering as much information regarding project specifications, security protocols and paradigms in software development, similar systems that had been previously developed, setup of team structure and creating all required documents. This will be the only sprint that will not have any development tasks and the rest of the sprints will be focusing on the development aspects only.

Activities

- Performing continual meetings with clients to ensure the project is fully understood
- Conducting team meetings to organise team and setup structure
- Creating SQAP (Software Quality and Assurance Plan)
- Creating Project Plan
- Performing meeting with the client to obtain feedback
- Creating SRS (Software/System Requirements Specifications)
- Creating an initial version of project design and architecture
- Creating Trello Board and allocate the team member accordingly to each task
- Creating a common Github repository where the team can upload their assigned tasks

Phase 2: Prototyping **Goal:** Planning of the basic system architecture, deployment of development environment, and prototyping functionalities.

Activities

- Performing meeting with the client to obtain feedback
- UI Block Chart and Layout
- Deployment of Development Environment
- Identification of libraries and needed tools
- Prototyping of core functions
- Mount the various file formats of the eforensic file and check its MD5 hash
- Check and verify the MD5 and SHA1 hash of file
- Find files that have been deleted or modified without the User and system's knowledge
- Submission of project planning documents
- Demonstration of progress

Semester 2 Phases The following are the semester 2 phases.

Phase 3: Minimum Features **Goal:** Development of the core functionalities of the system, updating of any previous documentation, creation of test cases, creation of project presentation video

Activities

- Creating the UI (User Interface) in both Graphical and also command line forms to access the system
- Creating the engine of the system that will be able to detect instructions provided and delegate to necessary functions functionality to Mount the various file formats of the eforensic file and check its MD5 hash
- Creating functionality to check and verify the MD5 and SHA1 hash of the file
- Creating functionality to find files that have been deleted or modified without the User and system knowledge
- Creating functionality to identify file content header and scan for keywords within the files
- Creating functionality to output a report with findings that include: the time zone of the user and computer, event timeline of a file (last modification time, modification details, file location manipulation) and a list of suspicious files and their details (name, Inode number, Mac dates, size, MD5 hash)
- Creating test cases to validate their functionality
- Creating proper documentation regarding their functionality and updating any previous documentation
- Creating Final presentation video to demonstrate the system

Phase 4: Optional Features **Goal:** Developing the optional requirements of the system with their own documentation and test cases, updating of any documents and demonstration video of the system

Activities

- Creating functionality for the system to create an eforensic file from a physical drive (like USB)
- Creating functionality for the system to perform memory forensics (eforensic file for memory, dumping of current memory into eforensic file, checking memory for suspicious activities)
- Creating functionality for the system to scan for suspicious attachments or keywords
- Creating test cases to validate their functionality
- Creating proper documentation regarding their functionality and updating any previous documentation
- Creating a new demonstration video of the system

4.3 Resources

To ensure that the system can be fully developed, access to the following resources is not only necessary but critical:

- Suitable Linux environment (Docker)
- Visual Studio or any equivalent editor
- Verbose documentation on Linux system
- Verbose documentation on several eforensic scripts/codes

Furthermore, it will also be beneficial for the team to have access to the following for research, testing and documentation purposes:

- Modern eforensic tools (like Super Timeline Scanner)
- eForensic tool kits (like Autopsy browser)
- Downloadable and Controllable Malware (like DarkCometRat, Mirai botnet, etc.)
- Cloud based repository to upload codes (like Github)
- Kanban style project tracking tool (like Trello)
- Web based learning management system (like Canvas)
- Communication and messaging systems (like Teams, Discord and Outlook)
- Shared Cloud storage (like One Drive)

Chapter 5

ORGANISATION AND STRUCTURE

A balanced matrix organisational structure will be used as it allows different members of the organisation to act as managers to a given functionality or activity in the project. Meaning that any given member can oversee an activity, while still being able to work on other parts of the project.

The key groups that are involved in this project are:

Project Manager – Responsible for liaising with all stakeholders, including bridging between the client and the development team. Also ensures that the activities and deliverables are being completed according to the project timeline.

Infrastructure Team – Responsible for creating and maintaining the various infrastructure that will be used throughout the project's lifecycle, including both development environments and repositories.

Development Team – Responsible for writing the software itself. Given the nature of the organisational structure, some members of this group will take on a more managerial role for certain functionality.

Quality Assurance Team – Responsible for ensuring that the software is functionally sound, as well as cooperating with the project manager to ensure requirements outlined by the clients are satisfactorily met.

Client – Responsible for setting out ensuring that the requirements and expectations of the software are met.
The table 5.1 describes the groups that will play a key role in the completion of a deliverable.

	Project Manager	**Infrastructure Team**	**Development Team**	**QA Team**	**Client**
Source Code			X		
User Manual	X		X	X	
Prototype		X	X	X	
Test Documents				X	
Binaries			X		
Presentation Video	X				
Design Documentation	X		X	X	
SRS	X			X	X
SQAP	X			X	
Project Plan	X				X

Table 5.1: Matrix Structure

Each role will play a leading role for the marked deliverables (and activities required to complete them), therefor the management load is distributed based on functionality, rather than everyone reporting solely to the project leader.

Chapter 6

RISKS

Rank	**Name**	**Occurrence Probability (H/M/L)**	**Severity (H/M/L)**	**Mitigation Strategy**	**Contingency**
1	Absent Team Member	Medium	High	6.2.1	6.3.1
2	Absent Client	Low	High	6.2.1	6.3.1
3	Major Skill Roadblock (team)	Low	High	6.2.2	6.3.2
5	Minor Skill Roadblock (individual)	High	Medium	6.2.2	6.3.2
8	Late Change of Requirement	Medium	Low	6.2.3	6.3.3
4	Project Data Loss	Low	High	6.2.4	6.3.4
7	Systems Acquisition Challenges	Low	Low	6.2.5	6.3.5
6	Inaccurate Time Estimates	High	Medium	6.2.6	6.3.6

Table 6.1: Risks

6.1 Risk Descriptions

Absent Team Member: This refers to the event of a Team member either leaving the team, discontinuing attendance of meetings, or not contributing to the production of the project plans, or software. This may produce issues such as an inability for the team to complete the tasks on time, higher team stress and conflict.

Absent Client: An absent client will create difficulties in completing the project primarily due to the lack of feedback when making progress on the project. The communication breakdown will result in an inability to define or clarify project requirements for the satisfactory completion of the objective.

Major Skill Roadblock: This constitutes a lack of knowledge or skill from the entire team that results in a particular feature not being implemented. If this feature is a dependency on other features that could present a major issue in project development and the completion of expected functionalities.

Minor Skill Roadblock: This presents similar issues to a major skill roadblock, except that it is only encountered by an individual on a particular task that was allocated to them. It is expected that through collaboration with the team, the roadblock will be trivial and can be overcome.

Late Change of Requirement: This risk occurs from poor communication with the client. And manifests if the client requests a feature change in the final weeks of development. The change may be reasonable or unreasonable.

Project Data Loss: This extends from the tools used to manage the development of the project. Each team member will be committing progress to a central repository. This has the potential to result in data loss or the merging of incomplete functionalities that break the program.

Systems Acquisition Challenges: This refers to issues that may arise when ensuring a standard operating environment for each team member and includes other external resources that may be required to complete the project but may take additional time to acquire. This could result in delays to expected completion deadlines. This also includes if particular tools are used that later are not suitable for the project.

Inaccurate Time Estimations: This pertains to the risk that the time estimates for each task that is off resulting in a failure to meet deadlines

6.2 Risk Mitigation Strategies

6.2.1: There is no effective way of minimising the likelihood of a team member or client having extended absences during the project. A variety of issues outside of our control could occur, whether that be the client having higher priority tasks, a team member exiting the unit, or having other events which in some way reduce their presence in the project.

However, the impact can be reduced, to reduce the impact of an absent client the team can ensure early definitions of project requirements and features, thus the team can continue to commit progress based on those early requests reducing the need for later client rendezvous.

To mitigate the impact of an absent team member, the team should clearly define the responsibilities of each team member and their expected contributions to the project. This combined with logbook entries should clearly state and account for each team member's contribution and allow for identification if a team member's production is below what is needed.

6.2.2: Skill roadblocks can primarily be mitigated through research and learning exercises. The initial strategy is to clearly define the needed knowledge and skillsets during the project planning stage. With these identified, each team member can catalogue areas of strength and missing or weaker skillsets. Using this information, team members will be able to upskill on the aspects of the project that they will need for satisfactory completion. This will reduce the likelihood of a skill roadblock occurring during the production phase.

6.2.3: To reduce the likelihood of unexpected or late changes to the scope or requirements of the project, there should be regular contact with the client, so they can communicate earlier rather than later any changes of use, scope, or functionality in the software. Furthermore, clear logs and minutes from those meetings should be kept ensuring there is accountability on both parties for the expected outcomes of the project.

6.2.4: Mitigating data loss will primarily be conducted using version control tools such as git. This will allow for revision tracking, contribution tracking and the ability to roll back in the event of a significant software breakage or data loss. In addition to this, it is ideal to maintain periodic backups that are stored in an offline medium if the git branches suffer a critical error or loss.

6.2.5: Analysis of the required systems should be conducted during the project plan to identify the needed systems before production starts. This way the team can ensure all resources are acquired for each team member's task to be completed successfully.

6.2.6: Team consensus on estimates for the tasks should reduce the possibility of the time estimates being incorrect.

6.3 Risk Contingency Plans

6.3.1: If a critical absence occurs, the project supervisor should be notified of the absence, and if the supervisor is absent then the unit convenor will be contacted. This will alert the necessary authority figures to the absence and provide corrective action to continue the project.

In the case of a team member's absence, their assigned contributions will be evaluated and redistributed among the present team members, according to available productivity and skill set. A meeting with the client may be conducted if necessary to evaluate and prioritise tasks if it is expected that some features will no longer be able to be met within the allocated time period of the project.

6.3.2: If an unexpected roadblock in knowledge or skill is encountered there are a few contingency plans that can be used to circumvent the issue. If the roadblock is within the scope of a single team member, the other team members can be contacted to collaborate to solve the challenge. It is expected that this will be all that is required to resolve the majority of roadblocks.

However, if a higher severity roadblock is encountered during production, where the entire team is struggling to resolve it the following remedial actions can be taken:

- A select individual can take further time to research the issue more thoroughly.
- The supervisor can be contacted to identify internal sources of help.
- The feature can be replaced on the backlog, to ensure that other features are not being delayed due to a single roadblock with one feature
- If no solution can be found, the client may need to be contacted to wither modify the requirements of the feature or remove it from the scope.

6.3.3: If a last minute change in scope does occur, the change should be evaluated to consider the following issues:

- What is the expected developmental life cycle of the change both with regard to development time and effort?
- Could this change be realistically completed within the time frame of the project, or would more time need to be granted?
- With the review of logs, had this change been previously requested or identified as a possibility.
- Depending on the answers to these concerns and discussion with the client the change will either be accepted, tentatively accepted or rejected, upon mutual agreement.

6.3.4: Data loss is still possible even when using version control tools, due to the mismanagement of branches, and the merging of incomplete or dysfunctional branches. In this event, offline backups should be used to replace the lost repository. The data loss (particularly if it is significant) should be accounted for. Logs of the work, including contribution records, and hours worked, should be documented. The cause of the loss should be analysed to identify the issue and a strategy should be made to prevent the issue in the future.

6.3.5: If an unexpected system or resource is needed during production, the team member can notify the team and plans can be made to either acquire it internally or request it from the client or supervisor if relevant.

6.3.6: Prioritisation will play a key element in mitigating the risk if it occurs. The plan will be to ensure that tasks that are critical success are completed first.

Chapter 7

Schedule

The project's activities can be grouped into phases, as per table 7.1.

Phase No.	Phase Start Date	Phase End Date
1	06-03-2023	07-05-2023
2	08-05-2023	28-05-2023
3	31-07-2023	01-10-2023
4	02-10-2023	29-10-2023

Table 7.1: Phases

Phase 3 & 4 will be conducted in two-week sprints according to these dates in table 7.2.

Sprint No.	Sprint Start Date	Sprint End Date
1	03-08-2023	17-08-2023
2	17-08-2023	31-08-2023
3	31-08-2023	14-09-2023
4	14-09-2023	28-10-2023
5	28-09-2023	12-10-2023
6	12-10-2023	26-10-2023

Table 7.2: Sprints

7.1 Project Timeline

The following two pages contain the project schedule displayed as a Gantt chart broken between semesters one and two.

7.2 External Dependencies

The following external dependencies may affect the schedule:

- Verifying Requirement document with client and supervisor.
- Getting supervisor's approval on the project plan.
- Validating expectations of users by meeting their requirements.

7.3 Assumptions

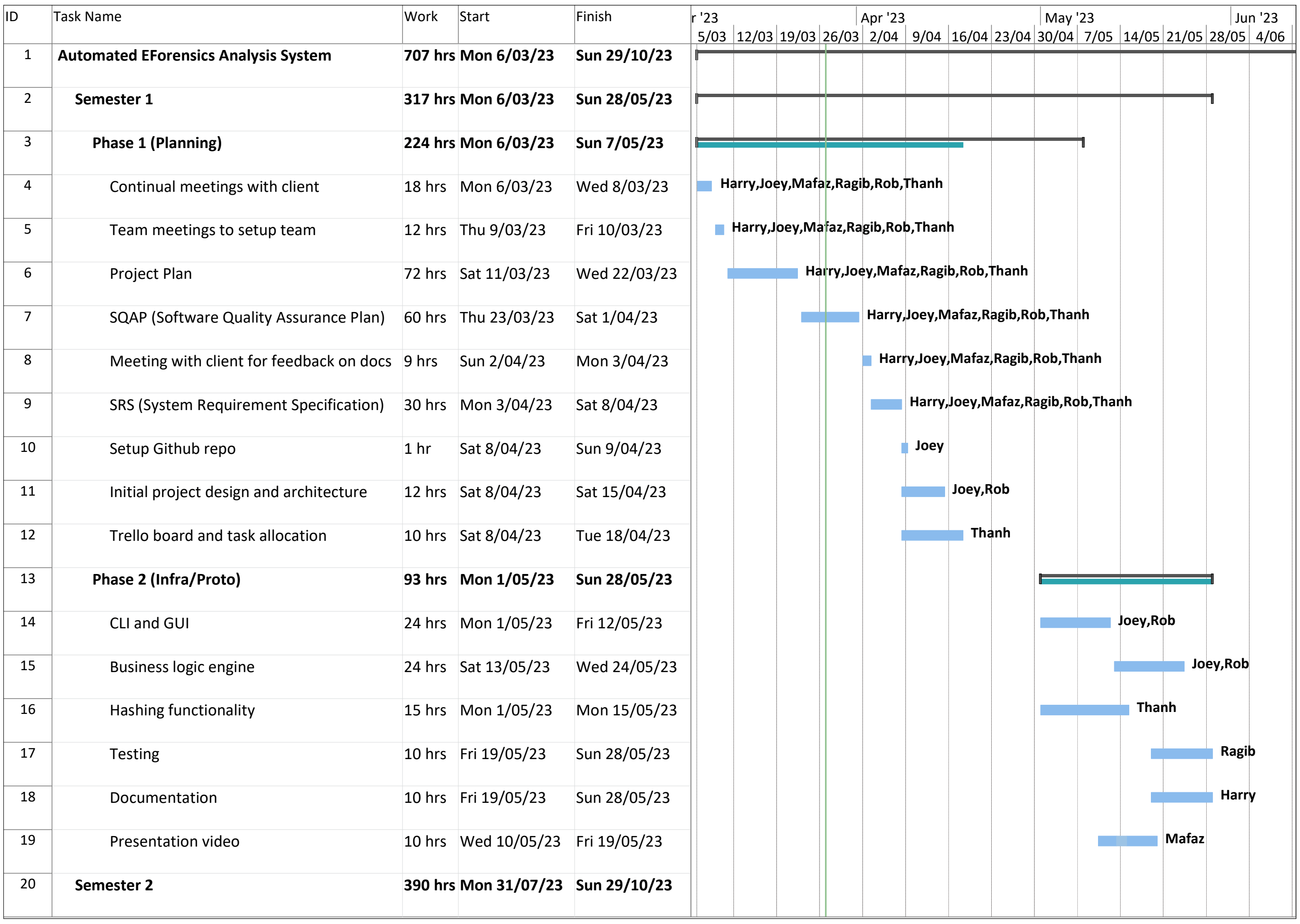
In the process of scheduling the work for this project we have assumed the following:

- The sprint scheduling process is dynamic, so the dates listed above are just estimates.
- The team members have sufficient skills to complete the tasks within their estimated time.

- The team will develop a sufficient understanding of the client's needs such that the team will not waste time on work that is not needed or needs to be redone.

7.4 Gantt Chart

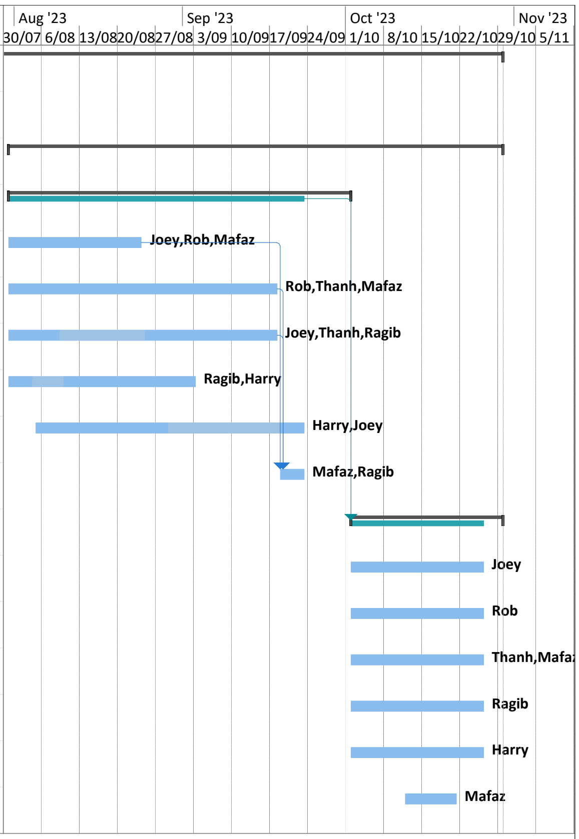
7.4.1 Semester 1 Timeline



ID	Task Name	Work	Start	Finish
1	**Automated EForensics Analysis System**	**707 hrs**	**Mon 6/03/23**	**Sun 29/10/23**
2	**Semester 1**	**317 hrs**	**Mon 6/03/23**	**Sun 28/05/23**
3	**Phase 1 (Planning)**	**224 hrs**	**Mon 6/03/23**	**Sun 7/05/23**
4	Continual meetings with client	18 hrs	Mon 6/03/23	Wed 8/03/23
5	Team meetings to setup team	12 hrs	Thu 9/03/23	Fri 10/03/23
6	Project Plan	72 hrs	Sat 11/03/23	Wed 22/03/23
7	SQAP (Software Quality Assurance Plan)	60 hrs	Thu 23/03/23	Sat 1/04/23
8	Meeting with client for feedback on docs	9 hrs	Sun 2/04/23	Mon 3/04/23
9	SRS (System Requirement Specification)	30 hrs	Mon 3/04/23	Sat 8/04/23
10	Setup Github repo	1 hr	Sat 8/04/23	Sun 9/04/23
11	Initial project design and architecture	12 hrs	Sat 8/04/23	Sat 15/04/23
12	Trello board and task allocation	10 hrs	Sat 8/04/23	Tue 18/04/23
13	**Phase 2 (Infra/Proto)**	**93 hrs**	**Mon 1/05/23**	**Sun 28/05/23**
14	CLI and GUI	24 hrs	Mon 1/05/23	Fri 12/05/23
15	Business logic engine	24 hrs	Sat 13/05/23	Wed 24/05/23
16	Hashing functionality	15 hrs	Mon 1/05/23	Mon 15/05/23
17	Testing	10 hrs	Fri 19/05/23	Sun 28/05/23
18	Documentation	10 hrs	Fri 19/05/23	Sun 28/05/23
19	Presentation video	10 hrs	Wed 10/05/23	Fri 19/05/23
20	**Semester 2**	**390 hrs**	**Mon 31/07/23**	**Sun 29/10/23**

7.4.2 Semester 2 Timeline

ID	Task Name	Work	Start	Finish	Predecessors
1	**Automated EForensics Analysis System**	**707 hrs**	**Mon 6/03/23**	**Sun 29/10/23**	
2	**Semester 1**	**317 hrs**	**Mon 6/03/23**	**Sun 28/05/23**	
20	**Semester 2**	**390 hrs**	**Mon 31/07/23**	**Sun 29/10/23**	
21	**Phase 3 (Complete)**	**245 hrs**	**Mon 31/07/23**	**Sun 1/10/23**	**13**
22	Deleted file detection	60 hrs	Mon 31/07/23	Thu 24/08/23	
23	File content header detection	55 hrs	Mon 31/07/23	Mon 18/09/23	
24	Report output	60 hrs	Mon 31/07/23	Mon 18/09/23	
25	Testing	30 hrs	Mon 31/07/23	Sun 3/09/23	
26	Documentation	30 hrs	Sat 5/08/23	Sat 23/09/23	
27	Presentation video	10 hrs	Tue 19/09/23	Sat 23/09/23	22,23,24
28	**Phase 4 (Optional)**	**145 hrs**	**Mon 2/10/23**	**Sun 29/10/23**	**21**
29	Forensic drive imaging functionality	25 hrs	Mon 2/10/23	Thu 26/10/23	
30	Forensic memory analysis	25 hrs	Mon 2/10/23	Thu 26/10/23	
31	Forensic email archive analysis	35 hrs	Mon 2/10/23	Thu 26/10/23	
32	Testing	25 hrs	Mon 2/10/23	Thu 26/10/23	
33	Documentation	25 hrs	Mon 2/10/23	Thu 26/10/23	
34	Presentation video	10 hrs	Thu 12/10/23	Sat 21/10/23	



Chapter 8

Budget

Table 8.1 lists the hourly rates for each team member. Table 8.2 describes the number of hours budgeted for each team member. Table 8.3 provides the estimated time to complete each activity.

Names	**Roles (Qualification)**	**Rate per Hour**
Mafaz Abrar Jan Chowdhury	Team Leader(Software Engineer Manager)	\$90.53
Joe Sutton Preece	Repository Champion (Senior Software Engineer)	\$76.65
Robert Findlay	Code Champion (Systems Engineer)	\$72.52
Thanh Nguyen	Sprint Master (Software Engineer)	\$49.40
Harrison Zito	Documentation Manager (Administrator)	\$26.51
S M Ragib Rezwan	End to End Champion (Software Test Engineer)	\$76.80

Table 8.1: Personnel Cost

	Phase 1	**Phase 2**	**Phase 3**	**Phase 4**
Team Leader	40	40	40	40
Repository Champion	40	40	40	40
Code Champion	40	40	40	40
Sprint Master	40	40	40	40
Documentation Manager	40	40	40	40
End to End Champion	40	40	40	40
Total Labour Expense	240h	240h	240h	240h

Table 8.2: Role hour Expense Estimates

Task	**Estimated Hours**
Phase 1	
Performing continual meetings with clients to ensure the project is fully understood	18
Performing continual meetings with clients to ensure the project is fully understood	12
Creating Project Plan	72
Creating SQAP (Software Quality and Assurance Plan)	60
Performing meetings with clients to obtain feedback	9
Creating SRS (Software/ System Requirements Specifications)	30
Creating a common Github repository where the team can upload their assigned tasks	1
Creating an initial version of project design and architecture	12
Creating Trello Board and allocating the team member accordingly to each task	10
Phase 1 Total:	224
Phase 2	
Creating the UI (User Interface) in both Graphical and also command line form to access the system	24
Creating the engine of the system that will be able to detect instructions provided and delegate to necessary functions	24
Creating functionality to Mount the various file formats of the eForensic file and check its MD5 hash	15
Creating test cases to validate their functionality	10
Creating test cases to validate their functionality	10
Creating a Final presentation video to demonstrate the system	10
Phase 2 Total:	93
Phase 3	
Creating functionality to find files that have been deleted or modified without the User and system knowledge	60
Creating functionality to identify file content header and scan for keywords within the files	55
Creating functionality to output a report with findings that include: the timezone of user and computer, event timeline of the file (last modification time, modification details, file location manipulation) and a list of suspicious files and their details (name, Inode number, Mac dates, size, MD5 hash)	50
Creating test cases to validate their functionality	30
Creating proper documentation regarding their functionality and updating of any previous documentation	30
Creating a Final presentation video to demonstrate the system	10
Phase 3 Total:	245
Phase 4	
Creating functionality for the system to create an eforensic file from physical drive (like USB)	25
Creating functionality for the system to perform memory forensics (eforensic file for memory, dumping of current memory into eforensic file, checking memory for suspicious activities)	25
Creating functionality for the system to scan for suspicious attachments or keywords	35
Creating test cases to validate their functionality	25
Creating proper documentation regarding their functionality and updating of any previous documentation	25
Creating a new demonstration video of the system	10
Phase 4 Total:	145
Overall Hours	707

Table 8.3: Task Time Estimates

Bibliography

[1] Indeed (2023) Junior Software Engineer salary in Australia - indeed. Available at: https://au.indeed.com/career/junior-software-engineer/salaries (Accessed: March 12, 2023).

[2] Indeed (2023) Senior software engineer salary in Australia - indeed. Available at: https://au.indeed.com/career/senior-software-engineer/salaries (Accessed: March 12, 2023).

[3] Indeed (2023) Linux engineer salary in Australia - indeed. Available at: https://au.indeed.com/career/linux-engineer/salaries (Accessed: March 12, 2023).

[4] Indeed (2023) Documentation manager jobs (with salaries) 2023 - indeed. Available at: https://au.indeed.com/q-documentation-manager-jobs.html (Accessed: March 12, 2023).

[5] Indeed (2023) Software test engineer salary in Australia - indeed. Available at: https://au.indeed.com/career/software-test-engineer/salaries (Accessed: March 12, 2023).

List of Tables