Automated Eforensics Analysis System

SWE40001
SOFTWARE ENGINEERING PROJECT A
SEMESTER 1, 2023
SQAP

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	20-03-2023	All Authors	Initial draft of draft created.

Acronyms

AEAS	Automated E-forensics Analysis System
ASAP	As Soon as Possible
DSDIR	Detailed System Design and Implementation Report
GUI	Graphical User Interface
IEEE	Institute of Electrical and Electronics Engineers
SADRR	System Architecture Design and Research Report
SQAP	Software Quality Assurance Plan
SRS	Software Requirements Specification
Ver.	Version

Contents

Chapter 1

Introduction

Author	**Student ID**	**Role Semester 1**
Mafaz Abrar Jan Chowdhury	103172407	Team Leader
Harrison Zito	103591785	Documentation Champion
Joe Sutton Preece	102568393	Repository Champion
Thanh Nguyen	101607169	Sprint Master
Robert Findlay	103613414	Technical Lead
S M Ragib Rezwan	103172423	End to End Champion

Table 1.1: Authors

1.1 Purpose

This document outlines the policies and procedures that members of Team 33 will follow to achieve an overall high standard of quality for the project titled “Automated e-Forensics Analysis System”, a data analysis system for Bad Security Inc. All team members are expected to adhere to the processes outlined in this document.

Chapter 2

Reference Documents

ISO 9001: Quality and Management System, Version 1.0 - 2015

The difference between a policy, procedure, standard and guideline (accessed 28-03-2023):

https://www.michalsons.com/blog/the-difference-between-a-policy-procedure-standard-and-a-guideline/42265

Chapter 3

Roles and Responsibilities

It is important for any organization to identify and clearly define the different roles to be taken on by different team members.

This chapter's primary focus is the discussion of these roles as well as the responsibilities they embody.

When reviewing these roles, it must be noted that these roles are not specifically bound to a single team member or person. Roles should exist separately from team members, allowing different members to flow freely between roles and shoulder a varied set of responsibilities. This ensures that in the case of failure or absence, the team can efficiently adapt to different situations.

3.1 Organizational Roles

This section lists and summarizes the roles within the project. We separate these roles into one of three categories: Meeting roles, formal review roles, and department lead roles.

3.1.1 Meeting Roles

These roles apply to formal weekly meetings, as well as to formal supervisor meetings. For informal meetings, which may occur spontaneously and without planning, these roles may be ignored. The practices for meetings are mentioned in 5.3.2.

Meeting Chair The meeting chair is responsible for leading the meeting overall. The meeting chair is also responsible for compiling items for the meeting agenda and presenting the agenda to the team at team meetings. The chair is also responsible for keeping time and ensuring that all agenda items are addressed. This involves keeping the discussion on the topic while ensuring that all team members' voices are heard. The Team Leader is expected to fulfil this role.

Scribe The scribe is responsible for recording the meeting minutes, formatting minutes and uploading the formatted version of the minutes to the group OneDrive folder. This role should be decided by the team at the start of every weekly meeting.

3.1.2 Formal Review Meeting Roles

The following roles apply for formal review meetings. Formal review meetings are covered in 5.3.2.

Moderator Plans the review and coordinates the review process.

Scribe Documents any issues or problems found during the review.

Author The creator of the work being reviewed.

3.1.3 Communication Roles

Client and Supervisor Liaison The Client and Supervisor Liaison acts as the single point of contact between the team, the supervisor and the clients. They are responsible for coordinating all incoming and outgoing correspondence between the team, the client and the supervisor. This responsibility also includes keeping these channels of communication isolated and unambiguous.

The Client and Supervisor Liaison will distribute all information received from the clients and supervisor to the team members. They are also responsible for setting up formal client meetings and formal supervisor meetings to ensure that communication is uninterrupted.

3.1.4 Champion Roles

A project of considerable size cannot be completed by one person alone. As such, the project must be split into sections and each section assigned to members to look after. It is the responsibility of champions to manage, oversee and supervise the sections assigned to them for the project's duration. Specifically, the champion is responsible for ensuring the quality of their section through the management of compliance standards and practices as outlined in this document. Champions should be the single point of contact for issues related to their assigned section.

Team Leader The team leader is responsible for promoting teamwork and cooperation over the duration of the project, resolving disputes and political issues within the team, and ensuring all members' voices are heard and considered when making policy decisions. They are responsible for monitoring project progress and ensuring that the project is completed successfully by maintaining clear channels of communication between members, supervisors and clients.

The team leader should also satisfy the responsibilities of the following roles: Client and Supervisor Liaison, Meeting Chair for formal meetings and Moderator for formal meeting reviews, except in the case where the team leader is also the author of the work.

Documentation Champion The documentation champion oversees the quality of administrative as well as technical documentation for the project. They are responsible for ensuring that these documents are consistent and complete before submission. This includes LaTeX formatting, document file naming and document version control.

Repository Champion The repository champion is responsible for managing the GitHub repository used for the project. This includes enforcing GitHub standards, maintaining directory structure, resolving certain merge conflicts and auditing commit messages. All issues with the repository should be directed towards the repository champion.

Sprint Master The sprint master is responsible for managing sprints once Phase 3 of the project has begun and sprints have been put into practice. The sprint master will maintain complete information on the state of the current sprint, specifically the state and progress of sprint deliverables. Any issues that will affect the timing and scheduling of the sprint should be directed towards the sprint master.

It should be noted that it is not the responsibility of the sprint master to resolve sprint related issues themselves, but rather to track and oversee the resolution of all such issues raised. For example, if a member alerts the team that there will be scheduling issues that will impede their ability to complete their assigned task, it is the responsibility of the sprint master to notify all members and confirm that another member gets assigned that task. The members to be assigned may be chosen by the team or by the team leader.

Technical Lead The technical lead is responsible for ensuring the quality delivery of code, maintenance of infrastructure, and tracking and assigning bugs and code issues. Their responsibilities include auditing members' code to ensure that uploaded code is standard compliant. They are also responsible for bug tracking and issue assignments. All queries regarding code styling, code infrastructure and module design must first be addressed to the Technical Lead.

End-to-End Champion The end-to-end champion is responsible for ensuring the quality and functionality of the application. They will be responsible for both usability testing and functionality testing. This will include ensuring that unit tests have been written by the associated developer and cataloguing the results of those tests. They need to understand the client's needs and ensure that the project is satisfying the objectives.

3.2 Responsibilities in Detail

3.2.1 General Team Member Responsibilities

The following lists the general responsibilities of team members:

- When a task is allocated to a team member, the team member must complete the task before the next formal team meeting. If the team member is unable to complete the task before the next formal team meeting, the team member must notify both the team and the team leader before the next formal meeting. This may be communicated via Discord.
- When a decision is reached on a specific topic in a formal team meeting, that decision is binding. Team members must not independently work against the binding item. Team members are free to work on additional material as long as they do not go against the binding items decided upon in formal team meetings.
- Individual team members are responsible for creating and logging their working hours on the timesheets on OneDrive.
- All members are to conduct themselves in a work appropriate manner, facilitating a healthy and supportive work environment.
- All members are required to maintain clear communication with the team.
- All members are required to follow all standards and processes described in this SQAP to the best of their ability.
- All members must make their best effort to attend all formal meetings and are to notify the team leader if they cannot attend.
- All members are to actively participate in group discussions during formal meetings and provide input to the product and processes.

Detailed Responsibilities by Role

The following lists some specific responsibilities as assigned to the roles discussed in 3.1.

Team Leader

- Responsible for coordinating all team meetings, except for informal team meetings.
- Responsible for the booking of weekly meeting rooms.
- Responsible for maintaining, finalizing and quality checking administrative documentation (together with the Documentation Champion)
- Responsible for motivating the team and tracking team progress.
- Responsible for monitoring and auditing individual work logs.
- Responsible for being a point of contact for issues/resolution.
- Responsible for liaising with the supervisor and the client.

Documentation Champion

- Responsible for creating and maintaining document templates.
- Maintaining quality and standards of documents.
- Responsible for providing support with documentation issues.
- Responsible for finalizing and quality checking the administrative documentation, along with the Team Leader.

Technical Lead

- Responsible for quality control of code artifacts.
- Responsible for tracking code progress.
- In charge of ensuring coding standards and best practices are met and followed, respectively, during the development process, along with the Repository Champion.
- Responsible for helping members with issues with the tech stack.
- Responsible for approval of changes in the tech stack.
- Responsible for resolving problems or inconsistencies with the tech stack.

Repository Champion

- Responsible for resolving issues related to Github.
- Responsible for creating and maintaining the repository.
- Responsible for monitoring commit messages and git issues.
- Responsible for maintaining file structures and location standards.
- Responsible for aiding with branching and merging.
- In charge of ensuring coding standards and best practices are met and followed, respectively, during the development process, along with the Technical Lead.

Sprint Master

- Responsible for maintaining the Trello board.
- Responsible for closing out resolved issues in Trello.
- Responsible for monitoring the progress of members with issues/tasks.
- Responsible for managing deliverables for sprints.
- Responsible for managing task delegation.
- Responsible for managing task deadlines for sprints.

End-to-End Champion

- Responsible for completing or delegating running of tests
- Responsible for ensuring that the main branch passes all tests.
- Responsible for building and maintaining testing standard documents.
- Responsible for ensuring that testing standards are being followed within the team
- Responsible for compiling test results for review.
- Responsible for ensuring team members take usability into account during development.
- Responsible for ensuring consistency of GUI.
- Responsible for ensuring that performance does not negatively impact usability.

Chapter 4

Documentation

All documentation will be contributed to by each team member. All documentation (aside from confidential self-assessments) should be stored on OneDrive to facilitate accessibility for each team member and supervisor. Documents that are to be formally submitted should be compiled in LaTeX and be signed by each team member, the supervisor and the client where relevant, as per 5.2.5.

4.1 Project Planning Documents

Project planning documents are written before development to identify critical project procedures, policies, resources, and expectations.

4.1.1 Project Plan

The Project Plan outlines the high-level expectations of the project, its objectives and deliverables, initial identification of required tools and standards, and Team structure. The project plan is a live document that will be revised as more details regarding the project are determined. A key responsibility of the project plan is to identify key milestones and outline the estimated schedule of completion.

Outline of Project Plan

Introduction Identification of project purpose and personnel

Terms of Reference Identification of objectives, scope, and acceptance criteria.

Establishment Identification of Process and Standards, the project and development environment and team's development skill requirements

Deliverables, Activities and Capital Resources Identification of tasks, project deliverables and required resources.

Organization and Structure Identification of roles and responsibilities.

Risks Identification of critical risk factors, mitigations, and contingencies.

Schedule Identification of key dates and expected milestones

Budget Outline of expected costs and time consumption

4.1.2 Software Quality Assurance Plan

SQAP aims to ensure that at the conclusion of the project, a sufficient quality of the software is achieved. Its role will be to outline the standard procedures of the team, including management policy, development practices, and testing methodology. This will also include outlines for conducting audits, code reviews, and how corrective action will be taken when problems are encountered.

SQAP Outline

Reference Documents Standard and Procedure Frameworks

Management Policies on enforcing adherence to practices and procedures

Documentation Outline of documents and reports

Standards, Practices and Metrics Determination of coding, communication, and management procedures

Review and Audits Methodologies of conducting and maintaining Reviews and Audits

Testing Outline of testing procedures and standards

Problems and Corrective Action Policies on resolving team conflicts

Tools and Methodologies Identification of mission critical tools for the project

Records Policies regarding the management of audits, meeting minutes and documentation.

4.1.3 System Requirements Specification

The purpose of the SRS is to detail the responsibilities, capabilities, and behaviour of the software. This is determined through the client brief for the project. From this, a detailed scope is created that will precisely outline the software's functional requirements.

SRS Outline

Introduction Determination of Software Purpose and Scope

Overall Description Details on project deliverables, acceptance criteria and required systems.

Functional Requirements Complete list of system features and functionalities

Non-Functional Requirements Complete list of quality metrics and thresholds

Interface Requirements Complete list of user, hardware, and software interface capabilities

4.1.4 System Architecture Design and Research Report

The SADRR outlines the high-level solution to the software based on the client brief and problem analysis. The document aims to determine the architecture of the software and conduct any research into existing solutions.

SADRR Outline

Introduction Outline the software to be developed and necessary research

Problem Analysis High-level analysis of the problem domain.

High-level System Architecture and Alternatives Description of the design of the software

Research and Investigation Documentation of research conducted for the solution domain.

4.1.5 Detailed System Design and Implementation Report

The DSDIR aims to cement the plans for the software. It determines in detail the system architecture, and functionality as well as module design, data structures and relations.

DSDIR Outline

Introduction Outline the software to be developed

System Architecture Overview High-level report on the SADRR

Detailed System Design Presentation of software design and justifications

Implementation Identification of key implementation decisions

4.2 Software Lifecycle Documents

These documents will be produced during the software development phases, which will be part of the sprints. This consists of three documents.

4.2.1 Module Plan

After assignments of backlog tasks to each member for the sprint, a module plan should be written for the item. This should outline the purpose of the module; its implementation plans and its structure. This may contain UML, Sequence diagrams or Pseudocode if relevant.

Module Plan Outline

Purpose Brief outline of the scope and why the module is necessary

Plan Outline of module functions and data structures

Tests Outline of needed tests

4.2.2 Code Documentation Comments

Within each code file, there should be comments to ensure an understanding of encapsulated functionality, parameters, and key implementation decisions.

Code Documentation Comment Types

Docstrings Outlines purpose and parameters of functions, data structures and objects

Inline Comments communicate key implementation decisions if required.

4.2.3 Sprint Review Document

At the conclusion of each sprint, there should be a review document that dictates completed modules, what tests were done and if any failed. Furthermore, this will contain code review information if needed.

Sprint Review Document Outline

Completed Modules list of completed Modules

Tests list of completed tests and results

4.3 Management Documents

Most Management documents will be produced every week, for the supervisor meetings. Others are produced irregularly as required by the unit outline.

4.3.1 Meeting Agendas

Meeting Agendas are formed throughout the week in preparation for the supervisor meeting. The agenda will contain a list of items as requested by the team to cover during the meeting. The meeting agenda is added to the OneDrive repository and submissions for topics close at the beginning of the meeting.

4.3.2 Meeting Minutes

These are collected during all meetings and compiled into a document. The meeting minutes outline the key topics covered and questions answered. It is not a full transcript. Along with every meeting will be a recording of the meeting which contains the full transcript.

4.3.3 Worklogs

Worklogs should be maintained as team members complete work at which they should update both the combined and individual worklog, the individual work logs should contain the item worked on, the amount of time spent and any issues that were encountered. Worklogs should be on OneDrive to be accessible to the supervisor.

4.3.4 Self-assessment reports

Self-assessments should be completed for the associated due date on canvas. It is the responsibility of each team member to complete and submit their own self-assessment. The document should be kept private.

Self-assessment report Outline

Summary summary of the self-assessment

Work Completed outline of tasks completed toward project

Mistakes Made outline of issues encountered.

Knowledge gained outlined of lessons learned and technical knowledge

Evidence supporting documentation of work done.

4.3.5 Audit Report

Audits will be carried out at the completion of key milestones, and upon request by a team member. This will be to validate the completion dates, workload of each team member and identify and resolve issues regarding technical aspects and team conflict. The audit should be available for all team members and the supervisor to assist in finding resolutions.

4.3.6 Deliverable Documents

Deliverable documents will be documents to be transferred to the client upon project completion.

4.3.7 System Documentation

This will include the complete details of the software, its architecture and functionality. The software documentation should include everything required for the client to set up the software internally without assistance from the team. This will pertain to the software environments, necessary packages and dependencies and code documentation for following developers and expansion.

4.3.8 User Manual

This should contain instructions and guides on how to use the software to achieve the project's objectives.

Chapter 5

Standards Practices and Guidelines

5.1 Purpose

Every project requires standards to serve as benchmarks. A standard is the consistent usage or configuration of specific tools and technologies. Adhering to well defined standards will help ensure software and overall project quality.

Similarly, projects require practices and guidelines to assist members in following a standard procedure and supporting the given standards. Having a set of practices and guidelines helps the team navigate the complex communication and development environments transparently.

The standards, practices and guidelines outlined in this section ensure that the team can effectively measure and ensure project quality. The process for assessment against these procedures can be found in the Reviews and Audits section.

5.2 Standards

The project will be held to the following standards to maintain quality. This project will be evaluated against these standards regularly to ensure they are upheld over the course of development.

5.2.1 Coding Standards

These standards set out the guidelines to be followed when writing the code. Preliminary standards are outlined below; however, the code standards should follow those that are recommended for the language chosen in development. For these standards, more descriptive names are preferred where applicable. Longer and more meaningful names that improve code comprehension are preferred over shorter names that may improve readability.

Variables and Constants Variables should follow the standard according to the language, such as snake_case or camelCase. Constants should take the same structure as variables however they must be in ALL CAPS.

Functions All functions should be named according to the language recommendation, either camelCase or PascalCase. All functions should be documented with short docstrings. Docstrings refer to multiline comments e.g. from typescript:

```
/**
* This is a docstring.
*/
function my_function_name():
# Function implementation.
```

The language used will determine the implementation of docstrings, and they must at least cover the purpose of the function and where it is meant to be used. Often a short sentence will be enough to document this.

Classes Classes must be named using the style according to the selected language, such as PascalCase, e.g. 'My-Class'. All Classes must also be documented with docstrings, following the same requirements as the functions above.

Modules Modules (I.e. files that serve as custom libraries/dependencies) must be named using the style according to the selected language such as snake_case or PascalCase. Modules do not require separate documentation, however, all functions and classes within the module must be documented as above.

Magic Numbers Code should avoid the use of 'magic numbers'. In this instance, magic numbers refer to numbers in the source code whose purposes cannot be understood at a glance. For example:

```
# Other code
total_height = add(5, 7)
# Other code
```

In the above, both 5 and 7 can be considered magic numbers. They should be replaced with:

```
HEIGHT_OF_SQUARE = 5
HEIGHT_OF_BORDER = 7
# Other code
total_height = add(HEIGHT_OF_SQUARE, HEIGHT_OF_BORDER)
# Other code
```

General Commenting Single line comments must leave a space between the start of the comment text and the comment character. Multiline comments must have the comment text start on a new line.

5.2.2 LaTeX Documentation Standards

LaTeX should be used for the final version of documents to be submitted. Documents typeset with LaTeX should follow the below standards.

General Formatting Document Class: Report (A4)
The cover page should consist of the Title, followed by a tabulated author list (name, id). The next page should consist of the table of content. Each topic will be a separate chapter with breakdowns using sections and subsections. There should be a new page for each chapter.
When referencing a figure, table or section the `\ref{tag}` should be used rather than referencing it statically.

Title Page Standard The title page will feature "Automated Eforensics Analysis System" as the primary title. The subtitles will contain SWE40001, SOFTWARE ENGINEERING PROJECT A, SEMESTER 1, 2023 each on a separate line. This will be followed by the name of the document below.

Table Standards Each table should be encapsulated within the `\begin{table}` tag. It should have a caption and label.

All tables should be centred on the page, this is achieved by encapsulating the `\begin{tabular}` with a `\makebox[\linewidth]{}`. Table content should be padded using
`\setlength{\tabcolsep}{10pt} % Default value: 6pt`
`\renewcommand{\arraystretch}{1.5} % Default value: 1‘` Before the tabular.

Figure Standards Each figure should be centred using `\centering` and include a caption and label.

Referencing An in document bibliography will be made using `\begin{thebibliography}[00]` and `\bibitem[id]` with associated `\cite{}` for in-text citations.

Appendices Appendices should appear at the end of their own chapter titled Appendices and be unindexed in using `\section*{name}`

Code Representation Multiline code blocks should be encapsulated with `\begin{verbatim}`, inline code will use the `\verb` tag.

5.2.3 File Name and Location standards

The following standards define the source code file naming principles to be followed for the project.

- All file and folder names will be lowercase.
- There should be no whitespace in filenames.
- File and folder names should only consist of alphanumeric characters, that is, no punctuation characters `(e.g. ', ', ?, !, ^ etc.)`, other than the dash character as discussed in the next point, should be used in file naming.
- The dash character ("-") will be used to delimit file and folder names when the name contains multiple words.

The above naming standard follows the well documented naming standard of kebab-case, which should be used for all files and directories within the \src directory (see 5.2.4).

Management/Administration files will be named using the following versioning format:
YYYY-MM-DD[ver.]_[Name of document]
Example: 2023-03-28a_project-plan.pdf

The date above should be the first date of publication, i.e. the date when the pdf is compiled and first uploaded to Github. The [ver.] indicates the version number and should consist of a single lowercase alphabet [a-z]. The version of the document should be updated every time it is changed and pushed again after initially being published to Github.

The underscore delimits the metadata and the document name. The document name itself should follow the kebab-case as before.

Published management and administration related files are to be stored within the \docs directory on GitHub (see 5.2.4).

5.2.4 GitHub Standards

Document Tree The following document tree describes the GitHub repository structure. Additional folders may be added at the discretion of team members after consulting with the repository champion.

```
\docs
    \final-presentation
    \sqap
    \project-plan
    \initial-client-email
    \assessment-criteria-agreement
    \srs
    \sadrr
    \dsdir
    \scs
\src
```

Repository Structure The repository should consist of a central main branch, which will contain fully tested, working production code. Maintainers should not be able to push directly to the main branch.

For each feature/functionality to implement, a new separate branch should be created. When the feature is completed, the branch should be merged with the main.

Branch naming The branch naming convention should consist of the prefix "feat" followed by the feature name in kebab-case, e.g. "feat-recover-hidden-files". For bug fixes (if an entire branch is required), the prefix should be changed to "fix" followed by the name of bug being fixed (again, in kebab-case).

Committing Commit messages will be authored in accordance with the guidelines provided by the Conventional Commits specification with commit types based on the Angular convention as suggested in the specification. https://www.conventionalcommits.org/en/v1.0.0/

Each commit message should contain a summary of what was changed in the commit, to a reasonable detail. Commit messages should not be vague or confusing. For example, "Fixed weird bug." is not a good commit message. A good example would be:

fix: stop GUI screens from changing background colour unpredictably. The issue was a dynamic variable in the background.tsx.

Git commits should include related changes. This can be achieved by creating multiple commits split by purpose to capture all changes made. A single commit must not be so large as to discourage peer review.

README.md The README file for the repository must contain instructions on how to run the application for debugging after pulling changes.

It must also detail how to deploy the application into an executable after development is complete.

5.2.5 Document Releases

When documents are to be submitted external to the team such as for the university or client, the document should be converted to a pdf through LaTeX and appropriately named, as per naming standards in 5.2.4.

5.3 Practices

Practices refer to the steps and procedures that should be taken to comply with the policy decisions made by the team. The practices outlined below should be followed as closely as possible to ensure quality control.

5.3.1 Communication Practices

These practices refer to the steps that should be taken to ensure that communication is clear, unambiguous and uninterrupted.

Team The following practices refer to communication that will take place internally within the team:

- The primary mode of communication between team members will be Discord for text messaging and virtual meetings, followed by face-to-face communication for formal and informal meetings when necessary.
- Only a single Discord server should be used for all internal project communication.
- Only the "general" Discord text channel should be used for all written communication. All other Discord text channels should be reserved for resource management and information distribution.
- Communication directed towards all members (via the '@everyone' tag) must be acknowledged by all members using reactions (e.g., the 'thumbs-up' reaction). Members must react as soon as the message is first read.
- Communication directed towards specific members (via the '@member' tag) must be acknowledged by the referenced member through either a text response or a reaction. Members must respond as soon as the message is first read. The response does not need to be a direct solution to the issue, simply an acknowledgment that the issue has been received.
- Members may ask questions or raise issues in the "general" text channel, which are not directed at any specific member of the team. In this case, the team leader, or relevant champion should respond to the query within 24 hours. If no member responds within 24 hours, the member who raised the issue must send the issue directly to the team leader, whereupon it should be added to the weekly agenda for the current week, to be discussed in the formal team meeting.

Client

- Communications with the client will be conducted primarily through the team leader unless otherwise agreed by the team.
- Such exceptions can be coordinated via any of the teams' communication channels such as Discord or any formal meeting.
- The primary method of communication will be emailed, supplemented by messages on Discord in the Bad Security Inc. server.
- If asynchronous messaging is insufficient, the team may arrange an informal meeting with the client on Discord.
- Meetings will be held regularly on Microsoft Teams to maintain adherence to the project timeline. This should occur at least once every three weeks unless deemed unnecessary by the team and client agreement.
- At the end of every sprint, a formal sprint review meeting should be held with the client to demonstrate the deliverable, unless deemed unnecessary by the team and client agreement.
- The team will strive to have as many members present for client meetings as possible and must have at least two members present.
- All Outlook emails to the client should CC the entire team.

Supervisor

- Formal contact with the supervisor will be conducted by the Team Leader, through formal supervisor meetings.
- Meetings will be held regularly on Microsoft Teams to maintain adherence to the project timeline. This should occur at least once every week unless deemed unnecessary by the team and supervisor agreement.
- Contact will primarily be through email and Microsoft Teams.
- All formal administrative and management documents should be sent for review to the supervisor through Canvas.
- All Outlook emails to the supervisor should CC the entire team unless they are of a personal nature.

5.3.2 Meetings

Team meetings are an essential part of ensuring clear communication, which leads to a higher quality of work. The team has divided meetings into five categories:

- Formal Team Meetings
- Formal Supervisor and Client Meetings
- Formal Reflection Meetings
- Formal Review Meetings
- Informal Team Meetings

General Meeting Practices These practices are to be followed for all meetings.

- All meetings must be between 30 and 60 minutes long. No individual meeting must last longer than 60 minutes. However, back-to-back meetings may exceed this time constraint.
- All meetings require an agenda.
- All meetings require meeting minutes to be taken.
- If a team member is required to be present at a meeting, however, they cannot attend the meeting, this must be communicated to the team leader ASAP, that is, as soon as the member realizes that there is a possibility that they may not make it to the meeting.

Formal Team Meetings The formal team meetings will take place weekly. All team members are required to attend.

- Formal team meetings will generally take place directly before the formal supervisor meeting.
- The goal of this meeting is to discuss the items that the whole team needs to discuss together to reach a policy decision. Examples include specific sections of management documentation that require input from multiple members of the team.

Formal Supervisor Client Meetings The formal supervisor meeting will take place weekly unless deemed unnecessary by the team and supervisor agreement. All team members are required to attend.

- The formal supervisor meeting will generally take place directly after the formal team meeting, and directly before the formal reflection meetings.
- The goal of this meeting is to discuss the current status of the project with the project supervisor, ask clarifying questions regarding the unit requirements and discuss other general administrative concerns regarding the unit.

The formal client meeting will take place on a tri-weekly basis (I.e., once every three weeks) unless deemed unnecessary by the team and client agreement. A formal client meeting will also take place at the end of every sprint to demonstrate the sprint deliverable unless deemed unnecessary by the team and client agreement. All team members are required to attend.

- The goal of this meeting is to discuss the progress of the project with the client, ask clarifying questions regarding the product and user requirements and demonstrate any deliverables completed by the team.

Formal Reflection Meetings The formal reflection meetings will take place weekly. All team members are required to attend.

- Formal reflection meetings will generally take place directly after the formal supervisor meeting.
- The goal of this meeting is to discuss the progress of the project within the team and to form an agenda and work schedule for the next week. This meeting also provides the team with an opportunity to reflect on the processes followed throughout the week and provide feedback to the team regarding performance.

Formal Review Meetings The formal review meetings will take place, when necessary, following the processes outlined in section 6. All team members are required to attend.

Informal Team Meetings The informal team meetings will take place spontaneously. A minimum of two members are required to attend. Informal team meetings will generally be used to discuss topics that require immediate attention, as well as to collaborate on completing individual team members' sections of their deliverables.

5.3.3 Worklogs

- Individual team members are required to update their worklogs to the shared OneDrive every week.
- The team leader should monitor and audit individual worklogs every week during the formal reflection meeting.

5.3.4 Coding practices

General guidelines The following are the general coding practices to be followed:

- Strictly follow official standards for the selected programming language, as outlined in 5.2.1.
- All source files must use a standard header template, which will be determined once the programming language is selected.
- All methods, fields and properties must have comments that follow the agreed upon standards in 5.2.1.
- Development must follow the architectural design, to ensure transparency in code.

Guideline on module plan outline Each module plan outline must be justified by a thorough analysis of the quality requirements of the module and applied design patterns. At the very least, a module outline plan should incorporate and separate the following components into directories in the library:

- Classes: all classes
- Utility functions: all utility functions
- Interfaces: all abstract classes and interfaces.
- Enumeration: all enumerations.
- Exception classes: all exception classes

5.3.5 Documentation Practices

During the writing of documents, MS Word will be used as it supports concurrency and collaboration. The development of documentation will only focus on the content of the document to ensure that all required topics are covered. MS word is chosen as it has a variety of tools available to support editing such as spell check and reword suggestions.

When writing new content for a document using MS Word, the text colour should be automatic, anything written in red is marked for deletion and will be removed during finalization and editing. If you need to write out a note that is not to be part of the content but is not to be deleted, then highlight or write with yellow. Nay references to other sections or tables should be written in blue.

Once the document has been approved by the team and signed as complete the document will be formatted using LaTeX. An appropriate template should be followed as per 5.2.2 for the type of document being produced.

Chapter 6

Reviews and Audits

6.1 Purpose

The reviews and audits section defines several procedures to build quality controls in the project through reviews and audits.

- A review is a meeting where one or more deliverables are presented to the client, supervisor or team for comments or approval. A review may be informal in the case where peer review is required. The meeting ensures that the correct deliverable meets the client's requirements and team standards. The review will be the primary method used before or after the formal deliverable's submission.
- An audit is a verification tool in which the lead roles of the deliverable will be examining the deliverable for any error, addition, or changes to the content. The audit will ensure all processes are met at an acceptable level to maintain product quality.

The standards, procedures and practices are in chapter 5, Standards and Practices.

6.2 Review/Audit list

6.2.1 Reviews

A formal review will be conducted for transitioning to or initiating a significant phase of the project that requires demonstration, reviewing, and testing of every completed deliverable. All formal reviews are conducted based on the team schedule.

An informal review is conducted based on the urgency of the deliverable.

6.2.2 Formal Review Process

1. All formal review meetings must use the following process:
2. Team Leader/client liaison will organize the time place for the meeting.
3. A review committee is selected and the specified roles are filled before each meeting whenever a regular member is not able to attend the meeting.
4. The moderator and author/s of the deliverables identify and confirm the review's objectives.
5. The moderator ensures that the review committee understands the objectives and the review process.

   Individual: the review committee will identify bugs, comment, and question anything about the deliverable. All these issues are recorded in the meeting minutes/personal notes for further review.

   Team: the review committee will return as a team to further discuss their findings about the deliverable in question. The team will determine the status of the deliverable and collect any additional questions about it.

   Team: If the review committee has deemed the deliverable unacceptable, the review committee will compile a listed improvement and issues to be given to the author of the work.

   Team: If the review committee has deemed the deliverable acceptable or completed, they be sent for further review with the client or supervisor.

6. The author of the work makes the required changes as specified in the list of improvements and issues from the review committee.
7. The moderator or review committee verifies that the actions required by the author have taken place at a satisfactory level.

6.2.3 Informal Review Processes

Code Review Any written code for this project needs to reach an acceptable level of code quality. A satisfactory level of code quality will go through multiple reviews below to ensure it meets the coding standard and specifications. If the code is deemed unsatisfactory, the author will be notified and provided with a list of improvements or changes to the code. The code in question is tagged as an issue on Trello and GitHub issues tab.

Peer review: Weekly inspections are carried out on all code commits by any peer developer before the next meeting. The Technical Lead will have to make the final decision on whether the code is satisfied or not. All peer reviews will need to consider the following aspects.

- Coding Standard
- Task Completion
- From each stage's detailed design, initial specifications are verified.
- Agreement upon any changes to specifications
- Satisfied with all required tests.

Client Review: Every two weeks, all completed branches are merged into the master branch on GitHub and notified to the client for testing and review. The client will determine the test site of the software, whether they want to test on their environment or demonstrate through the team's environment. The review and test of the software will match against the following.

- Deliverable timeline.
- Verified against specifications.
- Validate task completion.

Meeting The quality of the meetings is audited for each consultation. The audits will ensure the usage of the correct processes is acceptable. All meeting-related documents will be reviewed to ensure their quality.

All meetings will have an agenda. Before each discussion, the review committee will ensure the quality of the agenda. These agendas will consist of objectives, key issues, and questions.

Meeting minutes are documented by the Scribe and reviewed after each meeting. The reviews will validate that the meeting minutes were correctly written and fill in any missing information during the consultation.

If any meeting-related documentation is considered unsatisfactory, the review committee will further inspect the document. The review committee organized a list of improvements for the work's author.

Management Documents The documentation champion will review all management documents. Before the paperwork is released, the documentation champion will ensure all documents match the documentation standard in Chapter 5. If any report is found unsatisfactory, the review committee or supervisor shall provide a list of improvements. These improvements are noted in the meeting minutes or a feedback document.

6.2.4 Audits

Audits should be held regularly during all phases of the project's lifecycle to ensure that the processes put in place are being adhered to.

Coding Practices Audit Coding practices will be audited by the technical lead if a developer's code is unsatisfactory during a peer review. Failure to meet defined coding processes/standards will result in a list of improvements being generated and communicated to the responsible team member(s).

Communication Practices Audit Communications will be audited weekly by the team leader. Failure to meet defined communication processes will result in a list of improvements being generated and communicated to the responsible team member(s).

Documentation Practices Audit Documentation standards will be audited as part of before the submission of the documents by the Documentation champion. Failure to meet the documentation standards will be communicated to relevant team members with recommendations for improvement

Repository Practices Audit The repository practices will be audited as part of the end-of-sprint maintenance by the repository champion. Failure to meet the GitHub standards will be communicated to relevant team members with recommendations for improvement.

Review Practices Audit Review practices will be audited by the Team Leader on a case-by-case basis (If the review process seems too unfair). The team will inspect the review practices process and comment on how to improve the system.

Sprint Practices Audit Sprint practices will be audited as part of the end-of-sprint maintenance by the sprint master. Failure to meet the sprint standards will be communicated to relevant team members with recommendations for improvement

Testing and Usability Practices Audit Test practices will be audited weekly by the End to End champion. Failure to meet the testing/usability standards will be communicated to relevant team members with recommendations for improvement.

Chapter 7

Testing

Here, testing will be performed for both the CLI (Command Line Interface) and the GUI (Graphical User Interface), with the assistance of the client. These have been deemed acceptable by the team as they fulfil both the desires of the client and the baseline requirements of the software itself. Furthermore, all these tests must be performed on 3 different e-forensic images and only when they pass for all 3 cases would they be deemed successful enough to be handed over to the client. These will include various forms of testing including Unit Testing, Integration testing, System testing, etc. which have been noted in 7.1. All of this will ensure that the program is working as intended in providing the list of suspicious files (any file that has been modified or deleted without the users' or system's explicit permission/ knowledge)

The team will also perform some usability testing before its release to ensure that the UX (User Experience) principles of good practice (mentioned in SRS) have been maintained. As the project is heavily reliant in tracking and documenting suspicious files, the number of times when IO Exceptions can be thrown is quite large. Thus, whilst testing, the tester should ensure to document all error messages and general comments relating to this matter. Furthermore, the tester will also have a list of performance metrics to test and monitor (Ie CPU usage, memory usage, etc.) whilst performing their tests.

Once the team has made their best effort to document and resolve the issues that popped up during testing, the software will be sent to the client for feedback. Here, the client will also be provided with a feedback sheet which they will complete and return to the team. The feedback sheet will basically be used by the client to inform the team of any issues that they notice and will also be used as a process to detect and resolve any issue that might have remained. It will also be used by the client to confirm whether the performance level of the software is up to the level desired by the client.

7.1 Test Types and Rationale

Since several different types of testing are possible in this software, these have been noted in the table 7.1 (with their rationale) for ease of view:

7.2 Client Requirements

Since the overall goal of the team is to satisfy the client's needs, strong communication with them is crucial in producing a quality product that is both functional and maintainable. This will be reconfirmed in the SRS document (which will be verified and validated by the client)

7.3 Use case generation

Use cases will be provided by the team and verified and validated by the client. This will ensure that transparency is maintained between the client and the team and also allow the client to provide proper feedback regarding the functionality of the software.

7.4 Installation and User Documentation Generation

The software will be released in an executable or .exe format that will be provided to the client. This would reduce the hassle for the client as they wouldn't need to download anything beforehand to run the software.

The client will also be supplied with a comprehensive user document that details the GUI with various examples of how to use it (in the form of screenshots accompanied with verbose instructions). The client will be given the documentation of the code, alongside the design documents for each and every part of the software. This would not only ensure that the client understands the inner workings of the software but also allow them to maintain it by themselves for future projects.

Furthermore, after all the tests have been completed, the Client will also be provided with all the source code alongside the final build version of the system.

Test	Test Objective	Testing type	Rationale
1	Mounting of e-forensic file functionality	Unit Testing	It will be used by the developer to test the specific classes and pieces of the code to ensure they perform as expected, detect software bugs and errors, and ensure that system design and architecture for that functionality have been maintained.
2	Hash detection of image and suspicious files functionality	Unit Testing	It will be used by the developer to test the specific classes and pieces of the code to ensure they perform as expected, detect software bugs and errors, and ensure that system design and architecture for that functionality has been maintained
3	Detection of suspicious files functionality	Unit Testing	It will be used by the developer to test the specific classes and pieces of the code to ensure they perform as expected, detect software bugs and errors, and ensure that system design and architecture for that functionality has been maintained
4	Retracing of steps for file manipulation functionality	Unit Testing	It will be used by the developer to test the specific classes and pieces of the code to ensure they perform as expected, detect software bugs and errors, and ensure that system design and architecture for that functionality has been maintained
5	Generating report functionality	Unit Testing	It will be used by the developer to test the specific classes and pieces of the code to ensure they perform as expected, detect software bugs and errors, and ensure that system design and architecture for that functionality has been maintained
6	Timeline building functionality	Unit Testing	It will be used by the developer to test the specific classes and pieces of the code to ensure they perform as expected, detect software bugs and errors, and ensure that system design and architecture for that functionality has been maintained
7	Time zone detection functionality	Unit Testing	It will be used by the developer to test the specific classes and pieces of the code to ensure they perform as expected, detect software bugs and errors, and ensure that system design and architecture for that functionality has been maintained
8	UI button position and ease of view of the test (font size, colour, contrast, etc.)	Acceptance Testing	It will be tested by the team (ensuring that best practice has been maintained in terms of UX) and verified by the client
9	CLI functionality	Acceptance Testing	It will be tested by the team (ensuring that best practice has been maintained in terms of UX) and verified by the client
10	Linkage between the UI buttons in the front end and the commands in the back end	End to End Testing	It will be tested by the entire team to ensure that the linkage between the objects and flow of information and action is working as intended and defined in the project architecture and design
11	Partial system functionality (whenever new functionality is integrated into the system)	Integration Testing	It will be tested by both the individual developer responsible for the functionality and the entire team to ensure that the flow of action and information of the previous and newly added functionality does not come into conflict and instead works as intended. It will also be used to verify the design and architecture of the software
12	Full system functionality	System Testing	It will be tested by the entire team to ensure that the entire system is working as intended following the information defined in the project architecture and design

Table 7.1: Tests

Chapter 8

Problems and Corrective Actions

8.1 Personnel

If a personnel/s causes recurring issues that disrupt the team workflow, the Team Leader is notified and raises the issue as urgent. The Team Leader will consult with every member and gather information about the personnel's disruptive behaviour. Once the Team Leader has gathered the facts, they will recommend corrective action for the disruptive personnel to the team before administering the correction. Corrective action can include but is not limited to counselling, team reorganization, protocol changes and bringing the issue to the supervisor.

8.1.1 Corrective action process

Documentation of the correct action plan is required. The correction action plan's file name should follow this format: **DATE(FULL NAME)[ISSUE CATERGORY][STATUS]**. E.g.: 2023-03-23(Bob Job)[Coding][CLOSED]. The corrective action process in this paper has been modified from ISO 9001: Quality and Management System. The correction action plan should contain the following:

1. Define the problem.

   Who is causing the issue?

   What is the problem?

   Where did the problem occur?

   When did the issues occur?

   Why did the problem become an issue for the team?

2. Define the scope.

   Define the size of the problem.

   How often does the problem occur?

   What is the problem affecting?

3. Containment actions

   Actions are required to prevent further damage by the problem.

   Detection actions to prevent the same problem from occurring.

4. Identify the root cause.

   Lack of training or skill

   Communication issues

   Personal issues

5. Plan Corrective action.

   Create achievable and measurable action. Outline the steps to mitigate/stop the root cause.

6. Implement the corrective action.

   Provide evidence of corrective action implementation.

7. Team Leader will verify corrective action work.
   What was done?
   Describe if the situation has improved or not.

8.2 Work

8.2.1 Project major timeline

Automated e-Forensics Analysis System (AEAS) shall contain multiple phases. The core phases will be developed as follows: Planning, Prototyping, Minimum Features and Optional Features.

The parent project is called AEAS. The parent project will have multiple Versions (AEAS(Major version). (Minor version)), and each shall be a major milestone for development.

Within the parent project, there will be multiple product backlog items. Each product backlog item will have multiple minor versions. With each major version being one iteration. The implementation of the agile development model will use the following postfix included in each product backlog item's version to indicate the stage:

- "rqr" indicates the requirement stage.
- "dsg" indicates the design stage.
- "dev" indicates the development stage.
- "tst" indicates the testing stage.
- "dpy" indicates the deployment stage.
- "rev" indicates the review stage.
- "luh" indicates the launch stage

Each of these postfixes will represent lists on Trello. Each product backlog item will have its due date. The due date will match the end of a sprint's date depending on its dependency on other modules.

8.2.2 Stage-dependent tasks

Stage-dependent tasks and associated issues must be placed in the appropriate product backlog item. The naming convention of these product backlog item files should be [FUNCTION NAME] (Major version). (Minor version)-(postfix stage). (E.g. MOUNTING_DRIVE 2.0-tst indicating this file belongs to the test phase of mounting function second release.)

8.2.3 Task creation

The sprint master will convert the meeting minutes into tasks and allocate tasks to each member based on their skill set. Team members are responsible for their workload and divided into smaller tasks as long they meet the deadlines of the assigned task.

Team members are responsible for creating tasks for bug reporting or planning relating to their workload. Before submitting a new issue, team members must check if the problem post has been published/solved. Trello will record every task and its allocation to the team member. GitHub/Discord will be tracking a list of issues.

8.2.4 Task assignment

Within 24h of task creation, the work needs to be assigned. If a task cannot be allocated instantaneously upon creation, the team will decide which team member has the proper skill set to have a high chance of completing the task. If that team member comments that they cannot finish that task, it is the team's responsibility to complete it.

8.2.5 Task life

If the task assignment is deemed inappropriate, the assignee (assigned to the task) must respond within 12h. The assignee needs a proper reason why the workload is unsuitable for them. The team or relevant champion decides whether the justification is correct or not.

If the assignee's reason is justifiable, the team will provide an alternative task or assist the assignee. This alternative task can be swapping workload with another assignee or making the original task into smaller manageable tasks.

Champions are responsible for generating a solution that follows the appropriate standards and practices. The team will approve the solution and deliver it to the relevant team members. If there are no further issues with the corrective action or the problem, the issue case will be marked as "Resolved".

Champions responsible for the solution should record their time commitment in their work log and within the issue case.

After an issue is marked "Resolved", the respective champion is responsible for "closing" the correct issue case.

8.2.6 Issue Categories

GitHub will have software development issues categories within the GitHub issues tab and other non-software development issues categories will be placed within the discord server's channel called issues. All issues will be recorded on Trello for tracking purposes. Categories can be updated to adapt to the project's development; the following are the most up to date:

Software development issues categories: ("x" = Module name)

- Design - Module "x"
- Coding - Module "x"
- Testing - Module "x"
- Deploying - Module "x"
- Software Reviewing – Module "x"

Non-software development issues categories:

- Administration
- Audit - External
- Audit - Internal
- Client Liaison
- Documentation - DSDIR
- Documentation - General
- Documentation - Project Plan
- Documentation - SADRR
- Documentation - SD
- Documentation - SQAP
- Documentation - SRS
- Lecture
- Management
- Meeting - Client
- Meeting - Other
- Meeting - Weekly
- Meeting – Supervisor
- Presentation Preparation

- Research - Coding practices
- Research - Documentation
- Review - External
- Review - Internal

Chapter 9

Tools and Methodologies

9.1 Tools

9.1.1 LaTeX

All documentation that needs to be written LaTeX should be compiled using Overleaf. This ensures that the PDF output is consistent across systems, and the editor used is up to the individual, however, if using a rich editor, care should be taken to ensure that the LaTeX standards outlined in 5.2.2 are followed.

9.1.2 Git

All source version control is to be handled by Git, with the source code repository being hosted at https://github.com/. The Git source control provided by Visual Studio Code (see 9.1.4) should be used in most scenarios, as it provides an easy-to-understand interface, with all relevant information being provided graphically to the user. If additional flexibility is required (that is not provided by Visual Studio Code's Git source control), the Git command line interface may be used.

9.1.3 Trello

The product backlog will be managed using Trello. This is properly suited to our Agile development strategy, as backlog items can be moved between sprints. Additionally, the stage of development that feature is in can be tracked, i.e. development, testing, completion, etc.

9.1.4 Visual Studio Code

Visual Studio Code should be used for code editing, as it provides extensions with additional features for several programming languages and technologies, meaning that a single code editor can be consistently used for editing any necessary files (including but not limited to: .gitignore files, Dockerfiles, and source code for the program being developed).

9.1.5 Issues tracking

Issue tracking is handled by GitHub, accessed via our repository at https://github.com/. This allows issues to be tied to the source repository, including easily submitting pull requests to fix a given issue. Additionally, issues on GitHub can be given tags for better organisation.

9.1.6 Docker Container

All development will occur on team members' local computers. A common Dockerfile will be provided via the Git repository to allow a consistent development environment to be created across machines. The docker container will be running Debian 11 and will contain all the necessary libraries and programs required for the development of the software. Additionally, as the software has a GUI, the container will be configured forward graphical windows to the host machine.

9.1.7 Discord

Team meetings that cannot be conducted in person should be conducted on Discord. This allows for both voice and video communication, including webcam or screen sharing functionality. Discord provides a reasonable alternative to in-person meetings.

9.1.8 Microsoft Teams

Meetings with the supervisor and/or client that cannot be conducted in person should be held using Microsoft Teams. This provides similar functionality to Discord (described in 9.1.6), however, it has features that better facilitate 'official' meetings. Such features include scheduling a meeting for a given time – and integrating with the participants' Microsoft Calendar and automated recording and transcription of meetings.

9.2 Development Methodology

9.2.1 Scrum

As discussed in the Project Plan, the team will use an AGILE development methodology known as Scrum.

9.2.2 Product Backlog & Trello

To facilitate the use of the Scrum development methodology, the project requires a Product Backlog to be drafted and finalized. The Product Backlog must contain a full list of features and functionalities that the team intends to implement in the application. These were decided upon using the Project objectives and Scope detailed in sections 2.1 and 2.2 of the Project Plan. Not all features are required to be implemented, however, at the very least, the clients will receive an application with features satisfying the critical success criteria defined in section 2.3 of the Project Plan, and these features must be deemed to be of an acceptable quality under the acceptance criteria requirements mentioned in section 2.4 of the Project Plan.

The online tool Trello will be used to record the Product Backlog. A link to the Trello board containing this information is found here.

Other than Product Backlog, the Trello board consists of the Sprint Backlog list, which is discussed further in the next section, as well as lists for every stage of development as outlined in section 8.

9.2.3 Sprints

According to the Scrum methodology, development should take place in stages known as Sprints. In each sprint, a number of items from the Product Backlog (decided by the team) should be moved into the Sprint Backlog list. Over the course of the Sprint, the items will be developed and completed. At the end of each Sprint, at least one complete functionality of the application should be delivered - I.e. a working prototype of the application with the developed feature should be available for the Client to review. At the end of the Sprint, the Client is expected to review the delivered functionality for feedback.

Each Sprint will last 2 weeks. Each Sprint will consist of three distinct team meetings (along with normal development).

Sprint Planning Meeting: In this meeting, the team will decide which Product Backlog items to move into the Sprint Backlog. The team will then allocate tasks to team members. The formal team meeting on the first day of the sprint will be used for this purpose.

Sprint Check-in Meeting: In this meeting, the team will meet to discuss the progress of the different team members. Specifically, the team will discuss challenges in development and brainstorm solutions. The formal team meetings in between sprints will be used for this purpose.

Sprint Review and Retrospective: At the end of the sprint, the team will meet to discuss the achievements and failures of the sprint. Specifically, the team should list the Sprint Backlog items that were completed over the course of the Sprint, the Sprint Backlog items that were not completed, and the reasons why these items. The formal reflection meeting on the last day of the sprint will be used for this purpose.

9.2.4 Sprint Backlog Task Allocation

All members of the team should be included in the decision process, which determines the Product Backlog items to complete in each Sprint. This is done in the Sprint Planning Meeting.

As soon as a Product Backlog item is moved into the Sprint Backlog, the item must be broken down using a Work Breakdown Structure (WBS), and the total time required to develop the item must be estimated and recorded on Trello.

The total time for all items in the Sprint Backlog must not exceed 80 hours (10 hours per member per week).

The Sprint Planning Meeting should then cover the allocation of tasks to team members. Each task should be allocated to at least one team member. Tasks allocated to members must be marked by assigning the task to the relevant member on Trello.

Task allocation should play to members' strengths – rather than round robin allocation, the team members should decide which tasks best suit their skillset and pick tasks to complete accordingly.

Team members must also keep in mind the relevant time constraints when picking tasks to complete. No member must take on tasks exceeding the 10 hours of work per week limit.

If a Sprint Backlog item has not been allocated to any member, however, some members have the reasonable time resources and skillset to complete the task, the task should be allocated to the member with the least workload. Else, the item must not be included in the Sprint Backlog for that Sprint.

9.2.5 Modules

The terms product backlog item, sprint backlog item and module are used interchangeably within this document. All three terms refer to individual items of development that will get transformed into deliverables at the end of sprints.

9.2.6 Development Workflow

The following steps will be taken in a typical development workflow:

1. Team member is allocated a set of tasks from the Sprint Backlog
2. Team member designs and documents module plans for all assigned tasks and works with the team leader and sprint master to clarify the sprint deliverable for their assigned tasks. The module plans developed must include the various stages of development, including design and testing, as outlined in section 4
3. Team member pulls all recent changes from the main branch on the GitHub repository.
4. Team member creates a new branch for the first module plan.
5. Team member develops the first assigned task, making regular commits to the branch.
6. Once a team member is satisfied with the development, the team member opens a pull request to merge their branch into the main. The team member must select another team member to assign as a reviewer before making the merge request.
7. While waiting for feedback from review, the team member is free to begin work on the second module plan (on a second new branch).
8. The reviewer provides code feedback within 48 hours.
9. If there are code issues that need to be resolved, the team member must resolve these issues ASAP. The resolution of issues should be prioritized over working on new features.
10. Once all issues have been resolved, the team member must merge the branch into the main ASAP. Merging into production should be prioritized over working on new features.
11. Once the new branch has been merged into the main, the team member will continue working on tasks until either all new branches have been merged, or the sprint duration ends.

12. On the final day of the sprint, before the formal team meeting for the week, the end-to-end champion must ensure that the main branch passes all regression tests.
13. The main branch will contain completed sprint deliverables, which will be demonstrated to the supervisor during the formal supervisor meeting for that week, and to the client during formal client meetings.

Chapter 10

Records Collection, Maintenance, Retention

All meeting related documents (Ie Minutes, Agendas, notes, etc.), as well as working drafts of management documents, will be added and stored in the team's shared OneDrive repository where the team members and supervisor can easily access it. Before these documents are added to the repository explicit permissions will be taken from all the participants of the respective meetings.

All code related documents (Ie source codes, binaries, etc.), as well as published versions of the management documents, will be added and stored in the Team's Shared GitHub Repository where the Team, Supervisor and Client can easily access it.

Both types of documentation will be retained in their respective repositories for the duration of the project.

Chapter 11

Risk Management

The aim of risk management is to identify and prioritize risks that would prevent the successful completion of the project objectives. The impact of threats identified should be minimized or the likelihood of occurrence reduced.

Index	Asset	Threat	Vulnerability
0	Team Member	Decision Conflict	Lack of Management
1	Team Member	Absent Team Member	Lack of Time Management
2	Skill set	Skill roadblock	Lack of Research
3	Repository	Corruption	Lack of Training/ local hosting
4	Repository	Data Loss	Lack of Training
5	Scope	Change of Requirements	Poor Client Communication
6	Schedule	Inaccurate Time Estimates	Lack of Experience
7	Application	Incomplete functionalities	Unclear scope
8	Application	Functional Errors	Poor Design Documentation
9	System Environment	Incompatibilities	Using different OSs
10	System Environment	Version Conflicts	Local hosting on docker

Table 11.1: ATV

Index	Likelihood	Impact	Risk
0	Likely (4)	Moderate (3)	12
1	Probable (3)	Moderate (3)	9
2	Often (5)	Major (4)	20
3	Rare (1)	Severe (5)	5
4	Unlikely (2)	Severe (5)	10
5	Unlikely (2)	Moderate (3)	6
6	Likely (4)	Minor (2)	8
7	Unlikely (2)	Major (4)	8
8	Probable (3)	Major (4)	12
9	Rare (1)	Minor (2)	2
10	Rare (2)	Minor (2)	4

Table 11.2: LIR

Index	Strategy	Description
0	Retain	The risk will be taken on, but with clear management procedures, the decision should be resolved promptly.
1	Retain	Through clear delegation of responsibilities, the void left by an absent team member can be identified and tasks redistributed, with time management accounting for delays.
2	Share	Research and education are conducted before development to ensure confidence in the skill set of each team member, furthermore, if a team member encounters a roadblock, they will be able to collaborate with other team members to solve the issue.
3	Avoid	Through a repository, the code can be hosted on the cloud, ensuring there are backups locally and on the main repository. Through branches, the quality of code can be maintained on the main branch. Rollback features can avoid the issues of corruption by going back to the latest non-corrupted version.
4	Avoid	Again, through GitHub, and proper use of its system's data can be recovered through rollbacks if a merge overwrites portions of the repository. There would also be backups locally on devices.
5	Retain	The team does not have control of the client's requests, if this is to occur a meeting will take place to establish the predicted delays, and the necessities of the changes to determine whether they can be completed within the given time and budget.
6	Share	The team will negotiate reallocation of tasks if one task is going over the expected time and another under the expected time to balance the workload of each member.
7	Avoid	The scope and design are defined before development begins. This will be made clear and approved by the client to ensure that the application has the full set of functionalities at completion.
8	Avoid	Tests will be conducted periodically on code, including unit, system, and usability tests. There will be a dedicated testing champion to ensure that tests are kept up to date and confirm their status.
9	Avoid	Using docker the environment will be standard across systems. Docker is known to function on each of the team members' host OS.
10	Avoid	The docker file which defines the environment will be controlled through GitHub thus every team member will be using the same version with the same packages installed.

Table 11.3: Strategy