Cyber Incident Response Plan Objectives and Evaluation

System Quality Assurance Plan

Project Team - 02

Name	Position	Email	Phone
Aidhan Mitsopoulos	Scrum Master	103598809@student.swin.edu.au	0428770628
Habib Mustawafi	Product Owner	102053200@student.swin.edu.au	0466368916
Numil Fernando	Development Team Member	103517163@student.swin.edu.au	0406164700

Thomas Davis	Development Team Member	103203475@student.swin.edu.au	0449619570
Huy Tran	Development Team Member	102559614@student.swin.edu.au	0403204649
Zahin Un Nafi	Development Team Member	103539510@student.swin.edu.au	0481834630

Table 1: Project Team

Review history

Version	Date	Author	Comments
1.00	31/03/23	Team	Created initial draft document.
1.1	31/03/23	Aidhan, Huy, Zahin, Habib	Introduction, Management, Documentation, Standards, Practices, conventions and metrics,
1.2	31/03/23	Thomas	Testing
1.3	1/04/23	Aidhan, Huy, Zahin, Habib, Thomas	Standards, practices, conventions and metrics
1.4	2/04/23	Thomas, Huy, Aidhan	Format checking, minor edits

Table 2: Review History

Acronyms/Abbreviations

ASAP
As Soon as Possible

COB
Close of Business (5:00 PM)

DMO
Defence Materiel Organisation

DMS
Data Management System

GUI
Graphical User Interface

IEEE
Institute of Electrical and Electronics Engineers

JDK
Java Development Kit

JRE
Java Runtime Environment

Ver.
Version

SQAP
Software Quality Assurance Plan

SRS

Software Requirements Specification

SVVP

Software Verification and Validation Plan

Table of Contents

11.5 Risks with respect to the client

Chapter 1: Introduction

1.1 Author List/Roles

Author	Student ID	Role Semester 1	Role Semester 2
Aidhan Mitsopoulos	103598809	Team Leader / Supervisor Liaison / Inspector	Team Leader / Supervisor Liaison
Habib Mustafawi	102053200	Coding Champion/ Github Moderator	Coding Champion/Github Moderator
Numil Fernando	103517163	Coding Champion/Testing Champion	Coding Champion//Testing Champion
Thomas Davis	103203475	Usability/Quality Champion	Usability/Quality Champion
Huy Tran	102559614	Sergeant At Arms/ Coding Champion / Client Liaison	Sergeant At Arms/ Coding Champion / Client Liaison

Zahin Un Nafi	103539510	Documentation Champion /Inspector	Documentation Champion / Inspector

Table 3: Team Roles

1.2 Purpose

This document outlines the policies and procedures that members of Team 02 will follow to achieve an overall high standard of quality for Project C.I.R.P.A, a Cyber Incident Response Plan Analysts (CIRPA) for Retrospect Labs. All team members are expected to adhere to the processes outlined in this document.

Chapter 2: Reference Documents

1. Github Best Practices: http://Github.apache.org/repos/asf/subversion/trunk/doc/user/Github-best-practices. Html
2. Fine tuning Open AI model: https://platform.openai.com/docs/guides/fine-tuning
3. Pep 8 Python Coding Practice: https://peps.python.org/pep-0008/

Chapter 3: Management

3.1 Organisation/Roles

The following list contains currently identified Roles:

3.1.1 Meeting Roles

Chair

Performed by the Team Leader and is responsible for running/controlling the meeting as well as deter- mining and distributing the agenda for the meeting. Apologies and items to be added to the agenda from other team members are to be sent to the Chair prior to meeting.

Sergeant at Arms

Keeps an eye on proceedings and time spent on each item. Responsible for keeping the meeting on topic and on time. Their conduct must ensure that team members still have the ability to voice their opinions and ideas. Additional time can be utilised for discussion outside of the arranged meeting time.

Scribe

Records the meeting minutes and is responsible for circulation of meeting minutes. This role will be rotated through team members on a monthly basis as follows: Huy, Numil, Thomas, Habib and Zahin. The Chair is not expected to take minutes.

3.1.2 Formal Review Meeting Roles

Moderator

Plans the review and coordinates the review process

Scribe

Documents any issues or problems found during the review

Inspector

Examines the document/product for defects

Author

The creator of the work being reviewed

3.1.3 Champion Roles

Champion is a role that is directly responsible for ensuring the quality of a particular area assigned to them, as well as the compliance of standards and procedures of their sections of the project. They are not however responsible for completing the majority of the work. They are to delegate work, assist where appropriate and are the single point of contact for issues.

Team Leader is responsible for ensuring the team works effectively together to achieve successful completion of the project. The Team Leader typically chairs team meetings and is the supervisor liaison. They should be notified of any political issues within the team. The team leader is also responsible for monitoring the completion of work logs to ensure members are recording time spent, and contributing. They are also responsible for writing status reports where applicable and attending team leader meetings.

Documentation Champion is in charge of making sure all documentation is consistent, complete and to a high standard.

Github Champion is responsible for ensuring that the repository is used to its full potential and enforcing the Github standards. They are also responsible for maintaining directory structure and resolving any issues with the repository.

Usability Champion needs to ensure the product meets usability related non-functional requirements. They should understand the Client Requirements and ensure that any products are built with the client use in mind. They should assist with testing to ensure the GUIs are usable and intuitive.

The Code Champion is responsible for quality and on schedule delivery of code. They are to ensure other members are building and uploading code that compiles and is up to standard. They are responsible for ensuring delegated works are completed in a timely manner.

Testing Champion is responsible for running and reporting on test results. This includes Unit and functional testing. They should work with the Usability Champion to ensure that exceptions and errors are understandable and informative. They are to ensure tests carried out by other members are captured and recorded

3.1.4 Communication Roles

Client

The Client Liaison acts as the single point of contact between the team and the client. This allows coordination of all incoming and outgoing correspondence with the client and distribution to all team members. They are also responsible for setting up regular Client Meetings

Supervisor

The Supervisor Liaison allows the University and Supervisor to have a single point of contact for the team. However, other team members may contact the Supervisor for issues themselves. The role is typically filled by the Team Leader.

3.2 Tasks and Responsibilities

3.2.1 *General Team Member Responsibilities*

- If a team member is selected for a task, they will complete the task by the allocated time. If unable to complete a task in time, a member is to raise an issue prior to the deadline with the team leader.
- Meeting Actions are binding unless changed at a later meeting.
- Team members are responsible for the logging of their own time sheets.
- Members are to conduct themselves in an appropriate manner facilitating a healthy work environment.
- Members are required to maintain communication with the team.
- Members are required to follow all processes as described in the SQAP.
- Members must make their best effort attend all allocated meetings/workshops and are to submit an apology if they are unable to attend.
- Members are to follow all directives from champions.
- Members are to actively partake in group discussion and provide input to the product and the process.

3.2.2 *Champions*

Team Leader

- Responsible for the running of weekly team meetings.
- Responsible for the booking of a weekly meeting room if needed.
- Responsible for maintaining administrative documentation.
- Responsible for motivating and tracking team progress.

- Responsible for monitoring work logs and time spent on projects.
- Responsible for being a point of contact for issues/resolution.
- Responsible for liaising with supervisor

Client Liaison

- This champion is directly responsible for all communications with the client.
- Correspondence from team members must be relayed to/from client in a timely manner via client liaison.
- Minor/non-urgent communications are to be collated to avoid bombarding the client.
    - Any compiled versions that need to be tested will be sent via the liaison.
- Liaison is responsible for ensuring the client receives all relevant information for a test, as well as distribution of the test results supplied by the client to the team (bidirectional communication).

Usability and Graphical User Interface (GUI)

- 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.

Documentation

- Responsible for creating and maintaining document templates.
- Maintaining quality and standards of documents.
- Responsible for providing assistance with documentation issues.
- Responsible for maintaining documentation tools.

Code

- Responsible for quality control of code artifacts.
- Responsible for tracking code progress
- In charge of organising developer meetings to discuss progress and address any difficulties or concerns.
- In charge of ensuring appropriate workload is assigned for each developer.

- In charge of ensuring standards and best practices are met and followed, respectively, during the development process.

Github

- Creating and maintaining repository.
- Monitoring commit messages.
- Maintaining file structure and location standards.
- Providing assistance with branching and merging.

Testing

- Completing or delegating running of tests
- Reporting results of tests to team and on Github
- Building test documents
- Encouraging testing withing the team and compile results

Chapter 4: Documentation

4.1 Software Documents

4.1.1 SQAP

The SQAP is a plan written before any development that outlines all standard practices and procedures to ensure a quality process therefore help produce a high quality product.

4.1.2 SRS

A Software Requirements Specification will be developed to describe the behaviour of the proposed DMS as derived from client requirements. The SRS will be based on the IEEE 830 standard but will be modified to make it appropriate for our project.

A general outline of the document is as follows:

1. Introduction

1.1. Purpose

1.2. Scope

1.3. Definitions, Acronyms and Abbreviations

2. Overall Description

2.1. Product Features

2.2. System Requirements

2.3. Acceptance Criteria

2.4. Documentation

3. Functional Requirements

4. Non-Functional (Quality) Requirements

5. Interface requirements

5.1. System in Context

5.2. User Interfaces

5.3. Hardware Interfaces

5.4. Software Interfaces

5.5. Communication interfaces

6. References (if any)

4.1.3 Project Plan

A document to guide the building of the product. It will include a brief description of the project and why it should be built, what needs to be done to build the software and a timeline for when modules should be complete.

As more modules are mapped and more details are known about each item, the project plan is to be updated. It should also contain milestones where applicable and deliverable dates.

4.1.4 Self-assessment reports

A self-assessment report is to be completed by each team member each as per the unit outline that will provide evidence of work completed and self reflection. It will document knowledge and experience that has been gained during the process.

The following self-assessment documents are going to be delivered along with this project:

- SEPA Students Contribution Statement
- Personal Worklogs

- Peer Reviews

4.1.5 SADRR - System Architecture Design and Research Report

The SADRR is going to give an overview of the System Architecture Design and may contain the following sections:

1. Introduction

1.1 Overview

1.2 Definitions, Acronyms and Abbreviations

2. Problem Analysis

2.1. System Goals and Objectives

2.2. Assumptions

2.3. Simplifications (if any)

3. High–Level System Architecture and Alternatives

3.1. System Architecture

3.2. Other Alternative Architectures Explored

4. Research and Investigations

4.1. Research into Application Domain

4.2. Research into System Design

4.3. Research into technical platforms, languages and tools

4.4. Other Research

4.1.6 Detailed System Design and Implementation Report

The SADRR is going to give a detailed overview of the program design and may contain the follows sections:

1. Introduction

    1.1 Overview

    1.2 Definitions, Acronyms and Abbreviations

    1.3 Assumptions and Simplifications

2. System Architecture Overview

3. Detailed System Design (using Object Orientation or alternative)

    3.1. The Detailed Design and Justification

    3.2. Design Verification

4. Implementation

4.2 Management Documents

4.2.1 Meeting Minutes

- Will be collected at every meeting.
- Will be collected as either a .docx file and then converted to a pdf file.

- Formatted minutes will be released on the team github where a pdf version of the file will be posted no later than the following day's COB (Close of Business)
- A PDF copy of the minutes can also be emailed if requested, however, members are expected to find the minutes on the discord channel and complete their actions independently.

Chapter 5: Standards, practices, conventions and metrics

5.1 Purpose

Standards are essential for measuring and thus ensuring software quality. This section covers technical, documentation as well as process standards which guide the project's development and management. These standards mainly govern the output quality of each project's deliverable: libraries, applications, documents. In addition, they also serve as the development guidelines throughout the projects.

This section also includes practices that the development team shall adhere to, and will be assessed against.

These standards and practices are the basis to ensure and measure quality of the project's deliverables. The detailed procedure for assessment against the standards and practices can be found in the Reviews and Audits section.

All documentation (including the full Github repository) will be available for the client upon completion of the project for any future usage.

5.2 Standards

The following standards will be used as the basis for quality control in this project. They shall be adhered to closely throughout the software life cycle. Standards will be reviewed to ensure that they are being met.

5.2.1 Coding Standard

The following language-specific standard is used:

- Python PEP8 coding standard

5.2.2 Documentation Formatting Standard

All text documents will be written in MS Word to ensure that everybody can contribute.

5.2.3 Filename/Location standards

- All file and folder names will be lowercase. With the exception of the code folder where uppercase characters are allowed for the purposes of integrating with Microsoft Visual Studio's standards.

- There shall be no whitespace (spaces) in filenames.
- Management/Administration files will be named as follows "filename yyyy mm dd".
- Coding files shall be organised in folders in this structure \trunk\code\serpa\subproject\, \subproject is the programming unit
- Coding file shall be named in format of subprojectabbr namespace filename.extension, where namespace represents the subcomponents of a sub project.
- Management/Administration related files are to be kept within the docs folder.
- Multiple related files with similar content such as the meeting minutes are to be stored within an appropriately named encapsulating folder.

5.2.4 Document Releases

In the event that a document is released to an outside party; be it submission to the university or the client it must be:

- Converted into a static format such as a pdf
- Appropriately renamed with a revision number
- Moved to a release folder within its current folder

Each release will be named as follows: filename rxxx, where xxx is a number. The first release will be 100 and each additional release will be incremented by 10 i.e. filename r100 will be followed by filename r110.

5.3 Practices

The following practices will be used as the basis for quality control in this project. They shall be adhered to closely throughout the software life cycle. Practices will be audited to ensure that they are being followed appropriately.

5.3.1 Communication Practices

Client

- All contact will be made to the client through the client liaison champion.
- In the event that the champion is absent or on leave, alternative arrangements will be made (In advance if possible).
- Contact will be primarily made through email.
- Phone calls may be used in the event that emails can not provide enough detailed information.
- Meetings with the client shall be regular, taking place approximately every three weeks. However correspondence via phone or email should be more regular
- Meetings will always have a minimum of two team members present.

Team

- Email will be the primary method of communication, when face to face communication can not be conducted.
- Student email addresses are to be used in all email communications.
- Emailing of documents will not occur if the item can or is held in the Github with the exception of finalised meeting agendas and meeting minutes in PDF format only.
- Emails need to be checked daily.

- If a request for reply or acknowledgement of email receipt, members are to respond upon reading for the first time.
- Email communication will be kept to a professional standard.
- Emails can be used to confirm verbal contracts.
- Skype meetings are permitted if face-to-face meetings are unable to be arranged, but should be kept to a minimum.

Supervisor

- Formal contact with the supervisor will be conducted by the Team Leader.
- Supervisor will be present at one meeting per fortnight, at the request of the team leader (typically the Weekly Team Meeting).
- Contact will primarily be through email.
- Agreements with the supervisor will be confirmed via email.
- All emails to the supervisor should CC the entire team unless they are of a personal nature.
- If the team meeting is held out of business hours, then a separate meeting should be arranged with the supervisor. If this cannot be achieved, then at the very least a status report should be emailed to the supervisor to ensure they remain up to date with the progress.

5.3.2 Meetings

- Team meetings will be held on a weekly basis and will have a standard length 30 to 60 minutes.
- All team members are required to be present.
- If a team member can not be present, an apology needs to be communicated directly to the team leader As Soon as Possible (ASAP).
- All meetings require minutes to be taken.
- A meeting outside of the weekly team meeting does not need all members present; notes are required.
- Any meeting with the client requires at least two members present, again notes will be taken. Notes will distributed to client for confirmation.
- Notes will be stored on the Github

5.3.3 Worklogs

- Weekly update of Worklogs on GitHub.
- Team Leader to monitor Worklogs.
- Team Leader to monitor and maintain the project hours summary sheet.

5.3.4 Coding practices

General guidelines

- Strictly follow Python standards as outlined in 5.2.1

Guidelines on project structure

There shall be only one Visual Studio project that will implement the integration for the GPT Model and its GUI.

All major components are to be implemented by Python and training the GPT model.

User interface is to be implemented through the use of readily available open-source Python libraries, a possibility of which could be Gradio. This will be further explored during the development stages of the project to see what is suitable for our needs.

Guideline on components design

Each component design must be justified by thorough analysis into quality requirements of the component, applied design patterns or tactics.

It is recommended that to the very least, a component design should incorporate and separate the following component into directories in the library:

- Interface: include all abstract classes and interfaces
- Enumeration: all enumerations
- EventArgs: all subclasses of EventArgs if any
- Structures: all data structures if any
- Contracts: contract classes if any

Guideline on naming and namespaces

Appropriate namespaces will be automatically created by Visual Studio if folder structure is set up in the project.

Chapter 6: Reviews and Audits

6.1 Purpose

This section of the SQAP defines a set of procedures used to validate project deliverables and to verify team processes with respect to defined requirements and standards.

The purpose of validation is to ensure that the correct deliverables are being produced with respect to the client requirements and team standards. This is done through internal and external reviews.

The purpose of verification is to ensure that processes outlined in the SQAP are followed to ensure product quality. This is done through internal and external audits.

The standards, procedures and practices can be found in chapter 5, Standards and Practices.

6.2 Review/Audit list

6.2.1 Reviews

Reviews are held during all phases of the project's sprint cycles, towards the end of each sprint cycle to ensure high quality is present in all aspects of the project.

Formal Review Process

All formal review meetings must use the following process, a formal review is to be declared on a case-by-case basis:

1. A review committee is selected, and the specified roles are filled.
2. The Moderator identifies and/or confirms the review's objectives.
3. The Moderator ensures that all members of the committee understand the objectives and the review process:
    (a) Individual: the review committee will prepare to review the work by examining it carefully for potential defects.
    (b) Team: the review committee meets at a planned time to pool the results of their preparation activity and arrive at a consensus regarding the status of the document or standard being reviewed

4. Author of the work makes the required changes as specified by the review committee.
5. Moderator verifies that the actions required by the Author have taken place.

Informal Review

- Processing Code: Code quality is to be ensured through regular reviews as listed below. In the event that code is found to be unsatisfactory the results will be communicated to the relevant team member and raised as an issue.

- Peer review: Code commits shall be reviewed by a peer developer, assigned in the team meeting, as recommended by the Code champion, for the following aspects. Weekly inspection will be carried out on all commits by the assigned peer prior to the next meeting.
- Coding Standard
- Task Completion
- Verification against initial specifications, from each stage's detail design.
- Agreement upon any changes to specifications

Meeting

Meeting quality will primarily be maintained through audits of the correct process, but all meeting related documents will also be reviewed for quality.

Meeting minutes will be reviewed following the first meeting of each secretary against the standards. This is done alongside the formal acceptance of minutes at the conclusion of each meeting.

Agenda will be accepted prior to each meeting and formally reviewed prior to the following meeting.

Any documents found to be unsatisfactory will have results communicated to the secretary and raised as an Issue.

Management Document

Management documents will be reviewed against document standards prior to being finalised and released. In the event that the document is found to be unsatisfactory, a list of improvements will be generated and raised as an issue. One example of this is the feedback sheets provided by the supervisor.

6.2.2 Audits

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

Coding Practises Audit

Coding practices will be audited by code champions on a case-by-case basis (normally as a result of consecutive unsatisfactory peer reviews). Failure to meet defined coding processes will result in a list of improvements being generated and communicated to responsible team member/s.

Github Practices Audit

Github Practices will be audited as part of the routine maintenance by the Github champion. Failure to meet the prescribed practices will be communicated to relevant team members with recommendations for improvement.

Chapter 7: Testing

Testing will be predominately completed by team members remotely. Throughout the testing the client may be contacted and provide feedback / recommended changes to ensure their expectations are being met. The team is to perform as many tests on the releases before handing over to the client. Cyber incident response plans will be utilised and compared against to ensure the software is generating complete and satisfactory responses to the given prompts.

The team will also perform usability / function testing on the prototype prior to release. As the project relies on importing a large amount of cyber incident response plans and information, it is important to ensure the prototype will run smoothly without errors. As part of this testing, the tester is to document all error messages and general comments relating to usability of the GUI. Some examples are: processing time, readability of response and termination of processes correctly.

Once the team has made their best effort to test the program, have provided usable error messages / feedback and are satisfied with the release, it is to be given to the client for further feedback and recommendations.

The release should be accompanied with feedback sheets for the client to complete and return to the team so that issues can be identified, and ultimately, resolved. The client will also be expected to provide an acceptable performance level for the metrics used. For example how much detail and how accurate should the chatbot's response be?

7.1 Requirement

The overall goal of the team is to satisfy the clients needs when building / training the software. Therefore, strong consultation with them will result in a product that works, is usable and maintainable. The requirements outlined in the software requirements specification (SRS) shall be verified and validated by the client to ensure the product is suitable for deployment and use.

7.2 Use case generation

Use cases shall be validated and verified by the client with the assistance of the testing team. Sample outputs from the client will allow a basic understanding for the team, but ultimately the client will be responsible for communication specific uses of the software to ensure the team can adapt it to suit.

7.3 Installation and User Documentation Generation

The releases / prototypes will be deployed in the form of a python script and potentially an executable which provides a URL to the usable software. There is no installation as such, the user simply runs the executable / python script. All input files will be located as part of the GUI.

The client will also be supplied with a comprehensive user document that details the GUI /webpage and should have some examples of how to use it. The client will also be supplied documentation of the code. This is to assist with maintainability and use for future updates.

Once code has been finalised, and testing is completed, they will be supplied with all of the source code and a final build of the software:

- Tailored for a particular module, tests will be outlined within the module plans.
- Modules will be tested against a set of defined use-cases as agreed by the client.

Chapter 8: Problem reporting and corrective action

8.1 Personnel

If an issue arises with personnel the Team Leader is to be notified. The Team Leader will follow up on the issue to gather all the facts. Once known the Team Leader will suggest a corrective action. Corrective Action can include but is not limited to: counselling, team reorganisation, protocol changes.

8.2 Work

8.2.1 Project major timeline

There is one parent project, which is Project C.I.R.P.A./ Project C.I.R.P.A shall contain multiple sub-projects, which are the major deliverables that will be developed.

Project C.I.R.P.A will have multiple versions in its development, from its prototype down to its final release, with each being key milestones in its construction.

To implement agile spiral process model, the following naming conventions will be included into each product version to indicate the stage.

1. “Pln” the plan phase
2. “Dsg” indicating the design stage.
3. “Dvp” indicating the developing phase.
4. “Tst” Indicating the testing phase
5. “Dply” Indicating the Deploy phase
6. “Rvw” Indicating the review stage
7. “Lnch” Indicating the launch stage.

Release date of each version is the due date, scope is to be planned according to this due date. Progress of the current stage can be viewed by visiting “Road map” page.

8.2.2 Task creation

The scribe is to convert meeting’s actions to tasks and assign them to appropriate team members. The team member is responsible to divide the task logically into smaller tasks if necessary. This will include any task creations relevant for any of the ongoing deliverables.

Team members would also create tasks as appropriate: for bug reporting, planning or report research. Task creators must check for existing issues prior to creating tasks.

8.2.3 Task assignment

When a task cannot be assigned upon creation, the respective champion of the task must perform assignment within 24h of task creation.

8.2.4 Task life

Assignee (who is assigned to the task), must respond within 12h if the assignment is deemed inappropriate.

Resolver is responsible to ensure solutions are checked against the appropriate standards and practices prior to marking the issue as “Resolved”.

It is stressed that the resolver must entered the time spent on the task into the time-spent box before confirming as ’resolved’

After an issue is marked "Resolved", respective champion is responsible to formally/informally review the task (exception for trivial tasks), then mark the issue as "closed"

8.2.5 Issue Categories

Categories can be updated to adapt to the project's development, the following are most up to date:

- Administration
- Audit - External
- Audit - Internal
- Client Liaison
- Coding - Prototype
- Documentation - General
- Documentation - SD
- Documentation - SQAP
- Documentation - SADRR
- Documentation - SRS
- Documentation - DSDIR
- Documentation - PSAC
- Documentation - SCS
- Lecture
- Meeting - Client
- Meeting - Other
- Meeting - Weekly
- Presentation Preparation
- Research - Coding
- Research - Documentation
- Review - External
- Review - Internal
- Github Management

Chapter 9: Tools and methodologies

9.1 Tools

9.1.1 Issues tracking

Issue tracking tool is Jira. The tool can be accessed via a link that will be provided to the relevant people associated with the project. It provides a simple to use and feature-rich platform to track issues, progress and effort by the team.

9.1.2 Github

All commits and updates by team members shall be done GitHub. Jira Cloud can also be integrated with GitHub cloud to make for an easier work environment.

9.1.3 Visual Studio

Visual Studio 2022 (available on the microsoft website) is selected as the standard Python development environment.

9.1.4 Discord

If Discord meetings are deemed to be necessary, then all team members will need to download and install Discord and have access to a microphone and speakers.

9.1.5 Microsoft Teams

Microsoft Teams will be used to conduct meetings between the Development team and Client/Supervisor

9.1.6 OpenAI/ChatGPT

ChatGPT will be the AI model that is being modified to understand and identify the key elements of CIRP's being parsed into it. This model's scripts and GUI will be altered to fit the demands of our project plan.

9.2 Design Methodology

- Circular Scrum/Agile Lifecycle Model
- Lifecycle model utilises multiple iterations which suits our modular design.
- Each iteration will have its timeline specified in the project plan.
- Each phase will have its timeline defined in the module plan.
- Can have independent modules being processed concurrently if practical.
- The spiral model is endorsed by the client, and the shared understanding of the methodology will assist in communication on the topic of project planning.
- The model treats each iteration as a new project plan modified to include the following phases:
    - Requirements: Determining requirements for current iteration of the project
    - Design: Building a project plan for this iteration including timeline, work allocation, and risk identification and management.
    - Develop/Test: Programming, prototyping, documenting, verifying, and validating as discussed in Design Phase

Chapter 10: Records collection, maintenance and retention

Minutes, Agendas and Notes from meetings are added to project team's Discord and Github Repository as described in Github Procedures. Minutes and Notes will be added following approval by meeting participants.

All documentation will be retained in the repository for the duration of the project.

Chapter 11: Risk Management

11.1 Purpose

Risk management is undertaken to facilitate the creation of a product that is high quality, on time and delivers the scope specified by the client.

11.2 Categorization

For this project three major categories of risks have been identified:

1. Risks with respect to the work to be done.
2. Risks with respect to the management.
3. Risks with respect to the client.

In the following sections each of these categories have their major risks identified. For each risk, a description, a probability to occur, its impact and the preventative/(reductive) action associated are given.

Both the probability of a risk occurring and the impact of a risk if it does occur have been quantified as being low, moderate or high. Actions have been categorised as preventative and reductive; preventative actions aim to reduce the likelihood of risks occurring and reductive actions reduce the impact of risks if they do occur.

11.3 Risks with respect to the work to be done

- Corruption of repository
    - Probability: Low.
    - Impact: High resulting in loss of work.
    - Reductive Action: Weekly backups plus local checkouts reduce impact significantly.

- Design Errors
    - Probability: High.
    - Impact: High design errors would potentially increase production time and/or produce a deliverable not valid to client requirements.
    - Preventative Action: Rigorous design methodology prior to development.
- Time Shortage
    - Probability: High.
    - Impact: High, resulting in a loss of product quality, loss of functionality or delivered past the deadline.
    - Preventative Action: Rigorous design methodology prior to development including work distribution and conservative timelines.
- Illness or absence of team members
    - Probability: High.
    - Impact: Variable impact dependent on time in schedule.
    - Reductive Action: Shared understanding of work allows load to be distributed.
- Software non deployable
    - Probability: Low.
    - Impact: High, will be unable to provide clients with the modified ChatGPT model.
    - Preventative Action: Regular contact with client with minor releases to ensure that they can be deployed on the system.
- Technical Risks
    - Probability: Medium.
    - Impact: High, potential technical difficulties with the software, hardware or infrastructure that could result in delays.
    - Preventative Action: Implementation of a backup system and technical support team to handle any technical issues that may arise.
- Resource Risks
    - Probability: High
    - Impact: Medium, the risk of not having adequate resources to complete the project on time and within budget.
    - Preventative Action: Reassessment of project scope and timelines, reallocation of resources or outsourcing to complete the project.
- Scope creep risks
    - Probability: Low.
    - Impact: Medium, would result in increased production time.

- Preventative Action: Assessment of the impact of the scope change, and the implementation of a formal change management process to ensure the project remains on track.

11.4 Risks with respect to the management

- Illness or sudden absence of team leader
    - Probability: Moderate.
    - Impact: Variable impact dependent on time in schedule.
    - Reductive Action: Emergency meeting to be organised by Team Leader to elect temporary team leader.
- SQAP not suitable for our purposes
    - Probability: Low.
    - Impact: Failure to follow SQAP would reduce product quality.
    - Preventative Action: SQAP to be produced with full team input and cleared with a team supervisor.
- Team member leaves the team
    - Probability: Low.
    - Impact: High, their roles and responsibilities are no longer being fulfilled, and the number of manhours the team can provide in a given time is diminished.
    - Preventative Action: None. However, roles and responsibilities will need to be redistributed to the remaining team members.
- Communication risks
    - Probability: Medium.
    - Impact: Medium, the risk of miscommunication between team members and stakeholders which could result in misunderstandings or delays in project delivery.
    - Preventative Action: Clear documentation of project goals, timelines, communication processes and timely escalation of issues or concerns to management or other relevant parties.

11.5 Risks with respect to the client

- Changing clients' requirements
    - Probability: Moderate.
    - Impact: Moderate, increased workload and timeline issues.
    - Preventative Action: Rigorous discussion of requirements plus official SRS document early in project timeline.
- Client unavailable
    - Probability: Moderate.
    - Impact: Low, unavailability for questions and software releases
    - Reductive Action: Vital questions for clients to be communicated before they become critical.
- Client abandons project
    - Probability: Low.
    - Impact: High, no more work modules for the project to complete.
    - Reductive Action: Get as many modules and requirements off the client as possible.
- Miscommunication with Client
    - Probability: Low.
    - Impact: High, can cause unwanted Scope Creep and modification.
    - Reductive Action: Clear question asking and direction of meetings.