Swinburne University of Technology

SWE40001

Software Engineering Project

Semester 1 2022

Software Quality Assurance Plan

[SEP 28] Document Submission System

Review history

Version	Date	Author	Comments
1.0		All Authors	Created initial draft document
1.1	31/03/2022	Adrian Sim Huan Tze	• Formatted document. New font styles added. • Structured document with suitable headings and subheadings. • Added cover page and footer with page numbers.
1.2	01/04/2022	Adrian Sim Huan Tze	Completed Chapter 7 - Testing
1.3	02/04/2022	Jun Wee Tan	Complete and finalize Chapter 8 – Problem reporting and corrective action
1.4	07/04/2022	Jun Wee Tan	• Completed miscellaneous task (including word formatting, indentation, table of contents). • Reviewed all the document section
1.5	08/04/2022	Xin Zhe Chong	Completed Chapter 6 – Reviews and Audits
1.6	10/04/2022	Richard Ly	Add brief descriptions of the SRS outline in Chapter 4 – Documentation
1.7	10/04/2022	Adrian Sim Huan Tze	• Reviewed and revised the entire document. • Make adjustment to some headings and numberings. • Table captions have been added.
1.8	10/04/2022	Jun Wee Tan	• Final review on entire document
1.9	18/05/2022	Adrian Sim Huan Tze	• Final review on every section before resubmission

Acronyms/Abbreviations

• **SQAP**	Software Quality Assurance Plan
• **SRS**	Software Requirements Specification
• **GUI**	Graphical User Interface
• **SVN**	Subversion
• **STP**	Software Test Plan
• **SDLC**	Software Development Life Cycle

Contents

1. Introduction

1.1. Author List/Roles

Author	Student ID	Role Semester 1	Role Semester 2
Jun Wee Tan	101231636	Team leader champion Supervisor & Client liaison	Team leader champion Supervisor & Client liaison
Adrian Sim	101225244	Documentation champion Supervisor liaison	Testing champion Supervisor liaison
Xin Zhe Chong	103698851	Coding champion Supervisor liaison	Coding champion Supervisor liaison
Richard Ly	103340644	Usability champion Supervisor liaison	Usability champion Supervisor liaison
Yovinma Konara	102426323	Testing champion Supervisor liaison	Github champion Supervisor liaison
Sandali Jayasinghe	102849357	Github champion Supervisor liaison	Documentation champion Supervisor liaison

1.2. Purpose

This document outlines the policies and procedures that members of Team 28 will follow to achieve an overall high standard of quality for Project Twenty-eight, a Document Submission system for the client Mr. Caslon Chua. All members of the team are expected to follow all standards and procedures outlined in this document to produce a project of high quality. The purpose of the document is for the client and team members to refer to all quality aspects and roles detailed in the document to provide a project of high quality.

2. Reference Documents

This Software Quality Assurance Plan document will reference:

- Swinburne Document Submission System Project Plan (Software Engineering Project group 28)
- WordPress PHP coding standards - WordPress Developer Resources. 2022. PHP Coding Standards | Coding Standards Handbook | WordPress Developer Resources. [online] Available at: <https://developer.wordpress.org/coding-standards/wordpress-coding-standards/php/> [Accessed 8 April 2022].

3. Chapter 3 - Management

3.1. Organization/Roles

The following list contains the identified Roles:

3.1.1. Meeting Roles

Chair

Undertaken by the Leader of the team, the Chair is responsible for hosting meetings, along with distributing the agenda for the meeting amongst participants. Any amendments or item additions to the agenda from other team members are to be sent to the Chair prior to the meeting. **Jun Wee** will be the chair of this project.

Sargent at Arms

Monitoring the legal proceedings and time spent on each task. The responsibility of Sargent at Arms is to keep meeting focus consistent in an appropriate manner. The conduct ensures no conflict arises between the team members when they are expressing ideas and suggestions. Project members Adrian and Xin Zhe will act as the Sargent at Arms throughout the whole software development.

Scribe

The Scribe is responsible for recording the meeting minutes and presentation of meeting minutes to team members. This role will be rotated through team members on a weekly basis as follows: **Adrian, Xin Zhe, Richard, Yovinma and Sandali.** The Chair is not expected to take minutes.

3.1.2. Formal Review Meeting Roles

Moderator

This individual will be responsible for detailing and planning out the review and managing the process of reviewing.

Scribe

The scribe will be responsible for detailing and recording any issues encountered during the reviewing process.

Inspector
The inspector will be responsible for examining the project solution for any defects.

Author
The author will be the individual that details the work that is being reviewed.

Reader
The Reader has an obligation to read out the documents created for the inspectors.

3.1.3. Champion Roles

Champion roles are responsible for making sure quality is consistent in a specific area of tasks allocated for the individual to work on. These roles are to ensure that the task work is produced under the required standards and processes for the work delegated. These roles will not be responsible for task completion, but rather for assistance. The Champion roles for this project are as follows:

Team leader
This champion roles will be responsible for keeping the team in unity and ensure that they work effectively together to attain a successful project solution. This role holds the responsibility of initiating team meetings, attending team leader meetings and resolving any inter-member issues in the team, along will maintaining completed worklogs of team members and creating status reports where required.

- Jun Wee Tan will take up the team leader champion roles throughout the whole software project.

Documentation Champion
This champion role will be responsible for ensuring documentation is consistent, completed and up to standard, including formatting of text and names of files.

- Adrian will take up the documentation champion in the first half of project development process.
- Sandali will take up documentation champion in the second half of project development process.

Github champion
Github champion is responsible for maintaining consistent quality in the repository; this includes ensuring that the directory structure and resolve any issues encountered.

- Sandali will take up the Github champion in the first half of the project development process.
- Yovinma will take up the Github champion in the second half of the project development process.

Usability Champion

This role should ensure high quality on non-functional requirements regarding usability. It is important that client requirements are fulfilled, along with cooperating with testing Graphic User Interface.

- Richard will take up take usability champion throughout the whole software project.

Code Champion

Code champion ensures quality maintenance and on-time delivery of code while also ensuring that the code uploaded is non-erroneous and up to standard.

- Xin Zhe will take up take coding champion throughout the whole software project.

Testing Champion

Testing champion contains the role of running unit and functional testing and reporting test results such as exceptions and errors in an informative. They also ensure tests that are implemented by other members are captured, recorded and up to standard.

- Yovinma will take up the testing champion on the first half of project development process.
- Adrian will take up the testing champion on the second half of project development process.

3.1.4. Communication Roles

The communication roles are described as follows:

Client Liaison

The individual acting as Client liaison is responsible as a single contact point between the team members and client where all incoming and outgoing communication is distributed amongst client and team members accurately. Furthermore, they are also responsible for scheduling meetings with the client.

Supervisor Liaison

The supervisor liaison, who is usually the team leader, allows the university and the supervisor to be in a singular point of contact, on behalf of the team. It is also important to mention that the other team members are able to contact the supervisor regarding any issues or doubts.

3.2. Tasks and Responsibilities

3.2.1. General Team Member Responsibilities

The team responsibilities are detailed as follows:

- Team members are expected to complete tasks allocated within the allocated time; if the task cannot be completed, the team member should notify the team leader prior to deadline.
- If a meeting is scheduled, all relevant members are expected to be present on time, unless if changed to a later meeting. If individuals cannot attend, they will be required to notify the team leader in writing.
- Team members are solely responsible for maintaining their worklogs and submitting them to the supervisor every week with consistent quality.
- Team members are expected to communicate and interact with each other such that it continues to facilitate a healthy work environment.
- Team members are expected to maintain regular contact in an appropriate form with other members.
- It is expected that the team members adhere to all processes described under the Software Quality Assurance Plan.
- Team members are expected to inform team leader and supervisor regarding any conflicts or issues within the team or project.
- Members are expected to follow the roles of champions where relevant.
- Members are required to engage in group discussions and provide their own input regarding the project and processes followed.
- Team members are expected to be up to date regarding all progress regarding the project.

3.2.2. Champions

Team Leader

- Responsible for running and scheduling weekly team meetings.
- Responsible for monitoring and motivating team progress
- Responsible for ensuring that team members submit worklogs on time
- Responsible for being in a contact point for any issues
- Responsible for cooperating with supervisor where necessary

Client Liaison

- Responsible for all appropriate incoming or outgoing communications between client and team (in appropriate amounts)
- Responsible for sending any compiled versions and all relevant information of solution to client for testing
- Responsible for distributing test results of compiled version received by client, to the other members of the team

Usability and Graphical User Interface (GUI)

- Responsible for checking that the team members take usability into account when developing project
- Responsible for ensuring usability consistency and in a positively impactful manner
- Responsible for communicating with client liaison regarding any information to be sent/requested by client regarding usability.

Documentation

- Responsible for monitoring quality standards for documents
- Responsible for giving feedback and assistance to team members regarding documentation
- Responsible for maintaining consistency in formatting of documents and file names

Code

- Responsible for monitoring code progress
- Responsible for ensuring high quality of code, standards and best practices are followed
- Responsible for organizing meetings with team members to tackle code issues and update progress amongst team members
- Responsible for allocating appropriate workloads to each member

Github

- Responsible for creating and maintaining repository
- Responsible for keeping track of commits made to repository
- Responsible for maintaining correct file structure and location.
- Responsible for helping team members with regards to branching and merging.

Testing

- Responsible for delegating run tests and ensuring completion
- Responsible for distributing test results amongst team members
- Responsible for creating test documents
- Responsible for motivating team members for testing and compiling results.
- Responsible for ensuring that all aspects of the project are tested in the expected scope

4. Chapter 4 - Documentation

4.1. Software Documents

4.1.1. SQAP

The Software Quality Assurance Plan will be a written document outlining all the procedures, methods, techniques, tools, and standards to ensure that the Document Submission System is of quality.

4.1.2. SRS

The purpose of a Software Requirements Specification is to ensure that the project; web-based application for analyzing submitted assignments is based on the client's; Sir Caslon Chua request.
The SRS document will be made up of the following:

1. **Introduction**
   - Purpose – The aim of the SRS and the target audience.
   - Scope – Defines what the Document Submission will do and how it is going to accomplish its task.
   - Definitions, Acronyms and Abbreviations – The terminology that will be used over the duration of the development period.
2. **Overall Description**
   - Product Features – Will scan submitted work for plagiarism and then produce a text-based report.
   - System Requirements – The necessary components in order for the application to be deployed successfully.
   - Acceptance Criteria – An outline determining how the Document Submission System project could be deemed a success.
   - Documentation – The materials such as manual guide and Git repository will be in this section.
3. **Functional Requirements -** The Document Submission System must be able to scan and report a text-based report.
4. **Non-Functional (Quality) Requirements –** Essentially accessibility, usability, durability and modifiability.
5. **High-Level System Architecture –** Shows how the system is going to work.
6. **Interface Requirements –** How it will interact with other components
   - User Interface – how the user will interact with the application.
   - Hardware Interface – how the computer peripherals e.g., plugs, cables etc. are used to connect to other devices.
   - Software Interface – how information is passed from one application to another.
   - Communication Interface – how the application will interact with another application e.g., HTTP protocols.
7. **References**

   Any materials referenced will be written in this section.

4.1.3. Project Plan

The written document will outline the details of the project including project requirements, scopes, timeline of software development schedule. The project plan will serve as a big picture understanding for the project team to understand the requirements and to complete the project. It can also be used to keep track of progress based on the estimated time to complete sprint/work. The project plan also documents significant milestones if applicable, deliverables, risk management and deadlines.
The project plan should be updated as the project progress with more modules mapping.

4.1.4. SDRR Plan

The development of the SDRR plan should conform to the requirement analysis phase of each iteration according to the Scrum SDLC methodology.
The SDRR plan should contain the following:

1. **Introduction**
    - Overview.
    - Definitions, Acronyms and Abbreviations – The terminology that will be used over the duration of the development period.
2. **Problem Analysis**
    - System Goals and Objectives
    - Assumptions
3. **High-Level System Architecture and Alternatives**
    - System Architecture
        i. Component-and-connector-view
        ii. Deployment view
    - Other Alternatives
4. **Detailed Design (using Object-Oriented or alternative)**
    - Detailed Design and Justification
    i. UML Class Diagram
    ii. Design Patterns
    - Design Verification
5. **Research and Investigations**
    - Amazon Web Services (AWS)
    - Text Analysis Tool
6. **References**

4.1.5. Self/Peer-Assessment Reports

In week 6 and week 12, a self-reflection report is to be written and completed by each team member. In the document, each team member will write down their experiences over the course of the project and provide evidence to add credence to their own report.
The report should contain the following:

- A summary of the work so far
- Evaluation of each team member
- A self-evaluation
- Areas of improvement
- Knowledge acquired
- Evidence

4.1.6. Audit Report

An audit report must be produced subsequent to carrying out an audit. The report will document the outcomes of the audit; any process that does not obey the SQAP or the process is followed and disciplinary actions.
There will be an external audit and internal audit.

4.2. Management Documents

4.2.1. Meeting Agendas

- Every Friday at 9AM, Team 28 will have a meeting with the supervisor to give a progress report.
- Each member is expected to report what have they accomplish during the week to the supervisor.
- It is expected that each team member will have completed their worklogs for the week prior to meeting with the supervisor.
- Team 28 will be meeting up with the client once every 2 weeks.
- It is expected that the team to update on the progress of the project to the client.
- The document for meeting agendas is to be written in Microsoft Word.

4.2.2. Meeting Minutes

- The meeting minutes will be handled by each team member on a weekly basis.
- Each team member will take turns writing up the meeting minutes.
- The written document will be written in Microsoft Word.
- Each team member is expected to find and use the meeting minutes files.

5. Chapter 5 – Standards, Practices, Conventions and Metrics

5.1. Purpose

Standards are vital for ensuring that the output quality of the deliverables matches their market counterpart as close as possible. This section covers the technical documentation as well as the process standards which will be used as guidelines throughout the project's development and management. These standards mainly control the output quality of each project's deliverable: Research Report, Prototype System, and Documentation.

This section also describes practices that the development team shall follow and will be assessed against.

These standards are the foundation to measure the quality of the project's deliverables. A detailed procedure can be found in the Reviews and Audits section for assessing the deliverables against the standards and practices.

All deliverables from the project will be available for the client upon completion of this project for future usage purposes.

5.2. Standards

The standards described in this section will be used as the basis for quality control in this project. These standards should be closely followed throughout the software life cycle. Standards stated here will be reviewed to ensure that they are being met.

5.2.1. Coding Standard

The following language-specific standards are used

- WordPress PHP coding Standard

5.2.2. Documentation Formatting Standard

Where possible, all text documents will be written in a consistent manner to ensure that they are easily merge-able.

- Customizing of the font is possible, but every member must agree on using the custom font.
- Indenting of the markup should be done to improve readability
- Justification of the text alignment must be applied to the entire document except for tables and figures.

5.2.3. Filename/Location Standards

- All file and folder names will be lowercase.

- There shall be no whitespaces in filenames, any delimiters shall be replaced with an underscore "_".
- Multiple related files with similar content such as the meeting minutes are to be stored within an appropriately named folder.

5.2.4. Github Standards

Any commits to the repository must contain a corresponding message briefing the changes made, so that when changes are fetched, the progression of changes are easily understood and interpreted. The standard is as follows:

```
git commit -m "insert description of changes done here"
```

In the event where committed documents, folders are to be reversed, the individual with the role of subversion champion needs to be informed and approval must be sought prior to change. Furthermore, all files must be closed from their respective editors before committing files to the repository.

Branching standards

As a standard to be followed, the codework and documents will not be committed to the main branch upon first completion. Instead, it is expected that the members submit their documentation to a separate branch named "/test".

The members will push and commit their daily workload to a dedicated branch for testing, where it will undergo testing by other team members. Upon passing, the workload will then be committed to the "main" branch.

In the event where the workload does not pass the test, the user is required to commit the workload to a separate branch with an appropriate message. After this, the work can be fixed and re-committed to the main branch.

5.2.5. Document Releases

If a document is to be released outside for an appropriate reason such as submission to university or client referral, it would be released under the format described below:

- Converted to PDF format
- Appropriately renamed with revision number
- Moved to a release folder within its current folder for tracking purposes

Each release will be named in this format: filename_vx.x, where x.x is a number. The first release will be 1.0 and subsequent releases will be incremented by .1 i.e., **filename_v1.0** will be followed by **filename_v1.1**.

5.3. Practices

The following practices will be utilized with the intention of maintaining and adhering quality standards in the project, throughout the software development life cycle. Practices will be audited to ensure that they are being followed appropriately.

5.3.1. Communication Practices

Client:

- Establishing a communication channel that is convenient for both client and team to communicate with each other (Microsoft Teams).
- A temporary client liaison will be elected for communication with client in the event where the actual client liaison cannot be present.
- Contact will be primarily made via online meetings using the agreed communication channel.
- Email or Microsoft Team's chat feature may be used in the event where the online meetings cannot be conducted.
- Meeting with the client shall be regular, taking place approximately every two weeks. However, messaging via email or Microsoft Teams chat should be more regular and does not have to be limited to once every fortnight.
- Meetings will always have a minimum of three members present.

Team:

- Discord will be used as the primary method of communication, on cases where face to face communication cannot be conducted.
- Student names are to be used in all Discord communications. User tags are not allowed.
- Discord and Email needs to be checked regularly.
- Discord communication will be kept to a professional standard.
- Discord can be used to confirm verbal contracts.
- Microsoft Teams or Discord voice chat 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 should be conducted by the Team Leader.
- Supervisor will be present at one meeting per week, at the request of the team leader.
- Contact should be primarily through email.
- Agreements with supervisor should be confirmed via email.
- All emails to the supervisor should CC the entire team unless they are personal.
- If the team meeting is held outside of business hours, an alternative meeting schedule should be arranged with the supervisor as soon as possible. If this cannot be achieved, a status update report should be emailed to the supervisor to ensure they remain updated with the team's progress.

5.3.2. Meetings

- Team meetings will be held on a weekly basis and will have a duration between 30 to 60 minutes.
- All team members' presence is mandatory.
- If a team member is unable to be present, an apology or a valid reason needs to be informed directly to the team leader as soon as possible.
- All meetings require minutes to be taken, which will be rotated amongst members every week.
- A meeting outside of the weekly team meeting does not need all members present; notes are required.
- Notes will be taken for any meeting with the client requires at least three members present. Notes will be distributed to the client for confirmation.

5.3.3. Worklogs

- Weekly update of Worklogs on OneDrive.
- Team Leader will be responsible for monitoring worklogs.
- Team Leader will monitor and maintain the project hours summary sheet.
- Worklogs should be completed on the day before the meeting with the Project Supervisor
- Worklogs should be reported to the Project Supervisor during the weekly meeting.

5.3.4. Github

- Temporary/intermediate files should not be committed.
- Files must be checked by Team Leader for approval before committing.

5.3.5. Coding Practices

General guidelines

- Strictly follow PHP standards as outlined in 5.2.1
- Keep the code simple, avoid using unnecessary "clever" code.
- Once a week meeting (30 minutes) to report progress or difficulties in development, could also be conducted after weekly team meeting. The aim is to ensure problems are known early and progress is comprehended by the whole team.

Guidelines on project structure

User interface is to be implemented by **Bootstrap** for better presentation quality.

Guideline on components design

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

Guideline on naming and namespaces

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

6. Chapter 6 – Reviews and Audits

6.1. Purpose

The purpose of this section is to verify that all the project deliverables are valid, and the team processes are carried out in accordance with the predefined standards and requirements. The activity of verification is to be conducted with the help of a set of procedures and measures.

Validation ensures that the deliverables produced conform to the client requirements along with team standards in mind. To build the right product! This is performed through both internal and external reviews.

Verification ensures product quality by following every process documented in SQAP. To build the product right! This is performed through both internal and external reviews.

The standards, procedures and practices can be learned in Chapter 5 – Standards, Practices, Conventions and Metrics.

6.2. Review/Audit List

6.2.1. Reviews

Reviews will be held throughout all phases of the project's life cycle.

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 from the team members, and the specified roles are filled based on their understanding of the system. 2 members are chosen per item.
2. The Moderator identifies and confirms the review's objectives.
3. The Moderator ensures that all members of the committee understand the objectives and the review process.
4. a. Individual: The review committee will review the work by examining it thoroughly for any potential defects.
   b. Team: The review committee meets at a planned time to gather the results of their reviews and arrive at a consensus regarding the status of the document or standard being reviewed
5. The author of the work makes the required amendments as specified by the review committee.
6. The Moderator verifies that the changes made by the Author as requested by the review committee are done.

Informal Review Process

Code

Code quality must also be ensured through regular reviews as listed below. If the code is found to be unsatisfactory, the results will be informed to the responsible team members and raised as an issue.

1. Peer review: Code commits shall be reviewed by a peer developer followed by the Team Leader, as recommended by the Code Champion. Weekly inspection will be carried out on all commits by the assigned peer prior to the next meeting.
    - Coding Standard
    - Task Completion
    - Consistency and Optimization
    - Agreement upon any changes to specifications
    - Verified against specifications
2. Client review: Every fortnight, all working branches are merged and sent to the client for testing and review against the following: (This will not require a meeting with the client)
    - Deliverable timeline
    - Verified against specifications
    - Validate task completion

Meeting

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

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.

Any documents found to be unsatisfactory will have results for the secretary and raise an issue.

Management Document

Management documents will be reviewed against document standards prior to being completed and released. If the document is found to be unsatisfactory, a list of improvements and constructive feedback will be generated and raised as an issue, such as feedback sheets provided by the project’s supervisor.

6.2.2. Audits

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

Coding Practices Audit

Coding practices will be audited by Code Champion on a case-by-case basis (normally because of consecutive unsatisfactory peer reviews). Failure to meet the pre-defined coding processes will result in a list of improvements being generated and the team member(s) responsible will be informed.

Communication Audit

Communications will be audited on a fortnightly basis by the Team Leader. Failure to meet the pre-defined practices will be informed to responsible team member(s) with constructive feedback and improvement recommendations.

Github Practices Audit

Github Practices will be audited as part of the routine maintenance by the Github champion. Failure to meet the pre-defined practices will be informed to responsible team members with recommendations for improvement.

7. Chapter 7 – Testing

Software testing will be mostly conducted in a STP written to meet the requirements of the software. The STP describes what kinds of testing are utilized in specific sections of the software. It may comprise management plan and testing methodologies with a summary of ***the test activities, resources needed to perform testing and test schedules*** which comply to the ***Agile SDLC*** in this project. The following subsections introduces and illustrates the implementation of the testing specifications with different tests.

7.1. Requirements

The ultimate project goal of the team is to make sure that the software is correctly built during the development phase, and it satisfies the client's requirements. Consistent meeting and catchup with the client yield a desirable product that matches with the client's descriptions. Software requirements which include ***functional requirements*** and ***non-functional requirements*** (also known as ***quality*** requirements) should be clearly documented in SRS and these requirements must be objectively verified and validated by the client before launching it to ensure that the software is ready to deploy.

7.2. Use Case Generation

The use cases should be validated and verified by the customer with the help of the test team. Client sample outputs will provide a baseline awareness for the team, but ultimately the client will be responsible for communicating specific uses of the software to ensure the team can tailor it accordingly. Due to the help of the knowledge of the internal

7.3. Software Testing

7.3.1. Unit Test

Unit Test is a test that should be reckoned and applied in all code to ensure that each individual unit performs the required functions and yields the desirable outcome with proper data. Unit testing is less difficult to be conducted as it is white box testing with the use of some frameworks, symbolic debuggers, and software stubs. With the assistance of the knowledge in interior working module, this testing aids in ensuring proper operation of each individual unit as the tests are usually generated with that knowledge.

7.3.2. Functional Testing

The objectives of these tests are to verify and validate the business functionalities and requirements as specified by the client. The test coverage includes creation of five multiple choice questions, web search in submitted solution, report analysis and data storing.

	Item	Description
1	**Test Objective**	Makes certain that the software has the correct functionalities which include creation of five multiple choice questions, web search in submitted solution, report analysis and data storing.
2	**Technique**	This testing will be conducted iteratively. Carry out each use-case flow or functions with both valid and invalid data, to verify the function according to the criterion below: ➢ Valid data given to the function yields the expected result ➢ Invalid data given to the function produces failure and error messages ➢ Front end data will be verified from database data verification
3	**Environment**	Will be conducted in Development and QA environment.
4	**Entry Criteria**	➢ Unit Testing is completed
5	**Exit Criteria**	➢ All detected faults have been addressed ➢ All planned tests have been performed
6	**Special Considerations**	➢ Testing team will check and verify the functionalities outlined in the project descriptions document. ➢ Any modifications on the functionalities may influence the testing plan and testing cycle. ➢ The time of retesting and bug-fixing may influence the estimation of the total testing time.

Table 1: Functional Testing

7.3.3. Integration Test

Integration Testing will be performed at the end. During this phase, the testing will be mainly focused on integration of different modules. It specifically validates the data flow from end-to-end to ensure correctness and smooth functioning of the system all in all.

	Item	**Description**
1	**Test Objective**	Makes certain that the software is properly working with all the integrations of various modules at the end.
2	**Technique**	Check for test points of integration between each finalized module: ➢ Verify data flow from end-to-end throughout modules ➢ Verify functional requirements throughout modules
3	**Environment**	Will be conducted in QA environment.
4	**Entry Criteria**	➢ All test points of integration between modules have been identified and test cases are ready ➢ Software development has been completed together with system testing activities for each module
5	**Exit Criteria**	➢ Get 100% test coverage status with the execution of all test cases.
6	**Special Considerations**	➢ Prior to the testing, testers must allocate adequate time to understand the design and data flow ➢ Testing team must have a well-organized coordination to perform such Testing ➢ The time of retesting and bug-fixing may influence the estimation of the total testing time.

Table 2: Integration Test

7.3.4. Regression Testing

Regression Testing will be performed on the entire software in various iterations when there are new changes on the code. Each of the testing module should have its own regression suite built and testing will be executed to assure that all the existing functionalities have no defects caused by the code changes.

	Item	**Description**
1	**Test Objective**	Makes certain that the existing functions of the software remain intact and defect-free from the implementation of the subsequent modules.
2	**Technique**	The earlier modules will have its own functionalities built in different iterations. The subsequent modules developed in later iteration are to be tested along with the existing functionalities.
3	**Environment**	Will be conducted in QA environment.
4	**Entry Criteria**	➢ Functional testing of current module has been completed. ➢ The existing modules have their own individual regression suite ready.

5	**Exit Criteria**	➢ Get 100% test coverage status with the execution of all test cases.
6	**Special Considerations**	None

Table 3: Regression Testing

7.3.5. User Interface Testing

UI Testing, also known as GUI Testing, will test the facets of software that user will interact with. This generally means testing the visual elements to make sure that they are functioning based on the specified requirements and conform to community or industry standards.

	Item	**Description**
1	**Test Objective**	To make certain that: ➢ Visual elements and characteristics such as position, size, state and focus conform to the standards. ➢ User Interface of the pages should follow the requirements. It would include display of controls, font, colour, size etc. ➢ User Interface of the Administrative page should be as per the requirements of the UI guideline of the project. It would include display of controls, font, colour, size etc.
2	**Technique**	Setup tests for the web pages to verify UI elements.
3	**Environment**	Will be conducted in Development/QA environment.
4	**Entry Criteria**	➢ Unit testing is completed. ➢ UI checklist, test plan and test specifications are prepared.
5	**Exit Criteria**	➢ All detected faults have been addressed ➢ All planned tests have been performed
6	**Special Considerations**	The time of retesting and bug-fixing may influence the estimation of the total testing time.

Table 4: User Interface Testing

8. Chapter 8 – Problem Reporting and Corrective Action

8.1. Personnel

Teamwork is an essential factor in the success of a software project. If any issue happens between the personnel, the team leader will call an emergency group meeting to address it. Any issue of personnel must inform the team leader in advance. Below the table is the assumption of the problem may happen in personnel and respective corrective action:

Problem Reporting	Corrective Action
Team members refused to attend any meeting and complete any workloads given	Draft email to team members and try to contact his/her in person.
Lack of in-person communication between remote project members	Conduct a constant daily/weekly meeting to follow each member's progress.
Team members do not know about their roles and are responsible in project development	Create work management tools such as Trello board to ideate and plan project together.
Communication issues between project members	Conduct small meetings for both team leaders and involved members.

Table 5: Personnel problem and solution table

8.2. Work

8.2.1. Project Major Timeline

This document submission system project will consist of **6 main phases**:

- The Planning phases
- The Analysis phases
- The Design phases
- The Implementation phases
- The Testing and Integration phases
- The Maintenance phases

Each phase is composed of a few modules to ensure completeness.

Our project will adopt the **agile development method with the scrum model** throughout the whole development process. From **Semester 1**, we will create **five sprint processes** (from 25 March 2022 to 3 June 2022). Each sprint will take up to *two weeks of working days*. The development phases of planning, analysis, and design development phases will be completed and reviewed within these sprints. Implementation development phases will also be started in these sprints.

We will create **two sprints** from the **winter break semester** (from 24 June 2022 to 5 August 2022). Each sprint will take up to *three weeks of working days*. Our team will continue the development phase of implementation during this sprints period.

As for **Semester 2,** we will create **six sprint processes** (from 5 August 2022 to 30 October 2022). Each sprint will take up to *two weeks of working days*. The development phases of implementation, testing and maintenance phases will be completed and reviewed within these sprints. Final verification and review are also done in this semester.

8.2.2. Tasks Dependency

Below is the task dependency table that indicates the modules under each phase and its dependency:

	Task Name	Dependency
Document Submission System Gantt Chart		
Planning		
1	Project requirement meeting	
Weekly client meeting		
2	Weekly client meeting 1	
3	Weekly client meeting 2	**2**
4	Weekly client meeting 3	**3**
5	Weekly client meeting 4	**4**
6	Weekly client meeting 5	**5**
7	Weekly client meeting 6	**6**
Weekly Supervisors meeting		
8	Weekly Supervisors meeting 1	
9	Weekly Supervisors meeting 2	**8**
10	Weekly Supervisors meeting 3	**9**
11	Weekly supervisors meeting 4	**10**
12	Weekly supervisors meeting 5	**11**
13	Weekly supervisors meeting 6	**12**
14	Weekly supervisors meeting 7	**13**
15	Weekly supervisors meeting 8	**14**
16	Weekly Supervisors meeting 9	**15**
17	Weekly Supervisors meeting 10	**16**

18	Weekly Supervisors meeting 11	**17**
19	Weekly Supervisors meeting 12	**18**
20	Plan for collaboration tools with all stakeholders	**8**
21	Identify key project personnel	**8**
22	Identify goals and background issue of the project plan	**8**
23	Identify risk associated with project	**8**
Analysis		
24	Produce Project Plan	**20,21,22,23**
25	Produce Software Quality Assurance Plan (SQAP)	**20,21,22,23**
26	Project Specific Assessment Criteria	
27	Produce Software Requirement Specification (SRS)	**20,21,22,23**
28	Produce Research report for Definition of Plagiarism	**2**
29	Software/System Design and Research Report (SDRR)	**31,32,33,34,35,36**
30	Test Plan	
31	Usability Assessment Plan	
32	Product Video	
System Design		
33	defined coding guideline (in SQAP)	**2,3**
34	User interface wireframe (in SRS)	**2,3**
35	Use case diagram with task description (in SRS)	**2,3**
36	All High-level software architectural design (in SDRR)	**2,3**
37	UML class diagram (In SDRR)	**2,3**

38	Entity Relationship Diagram (ERD) for database	**2,3**
39	Software work-breakdown structure	**2,3**
System Implementation		
Login module		
40	create front-end user interface	**33,34,35,36,37,38,39**
41	create login database schema and object-oriented class	**33,34,35,36,37,38,39**
42	create connection between user interface and database	**33,34,35,36,37,38,39**
43	Program user login form and validation	**33,34,35,36,37,38,39**
44	testing login module	**33,34,35,36,37,38,39**
Student Submission Module		
45	create submission front-end page	**33,34,35,36,37,38,39**
46	create respective database infrastructure and link both front-end and back-end	**33,34,35,36,37,38,39**
47	create object-oriented class	**33,34,35,36,37,38,39**
48	create upload, delete file function	**33,34,35,36,37,38,39**
49	create link MCQ question page	**33,34,35,36,37,38,39**
50	create system feedback function on submission	**33,34,35,36,37,38,39**
51	testing submission module	**33,34,35,36,37,38,39**
MCQ Question Module		
52	create MCQ question module front-end page	**33,34,35,36,37,38,39**
53	create respective database infrastructure and link both front-end and back-end	**33,34,35,36,37,38,39**
54	create object-oriented class	**33,34,35,36,37,38,39**
55	create generate question function based on user submission	**33,34,35,36,37,38,39**

56	testing MCQ question module	**33,34,35,36,37,38,39**
Admin Management module		
57	create 4 sub-pages (student submission, student list, question list, report analysis)	**33,34,35,36,37,38,39**
58	create respective database infrastructure and link both front-end and back-end	**33,34,35,36,37,38,39**
59	Create object-oriented class	**33,34,35,36,37,38,39**
60	create analysis function based on submission (sentiment, web search)	**33,34,35,36,37,38,39**
61	create function for generate user analysis report	**33,34,35,36,37,38,39**
62	testing admin management module	**33,34,35,36,37,38,39**
Testing and Integration		
63	integration and acceptance testing	**33,34,35,36,37,38,39**
64	bug fixing	
Maintenance		
65	User training	

Table 6: Task Dependency

Task Creation and Assignment

For task creation under each development phase, **all the project members** will be **the task creator and developer** to convert project requirement to tasks and divide the tasks into smaller tasks for every member.

Each **assigned champion of the task will also have to make sure the work allocation** for team member is equal and must be done within 24 hours upon the task creation.

8.2.3. Task Life

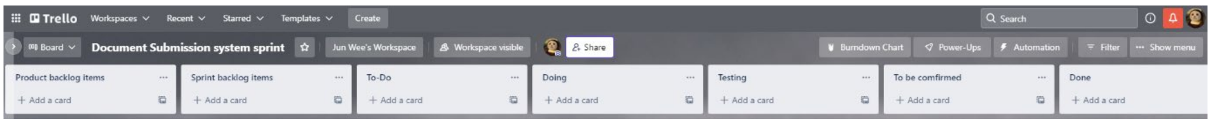
Team leader Champion will record every module task in project development in the project management tool, Trello board. Trello board will help us to track each task progression.

The Trello board will have **seven template cards**: Product backlog items, Sprint backlog items, To-Do, Doing, Testing, To be confirmed and Done. They will indicate the life of tasks.

First, the champion will gather all the extracted tasks from the development phase **in product backlog items section.** When the sprint date is started, the champion will move the respective task in the sprint to the **sprint backlog items section** by champions.

A team member will be assigned to a specific task and move the task to the **To-Do section** when they intend to do it. When a team member is doing the task, they should move the task to the **Doing section**. After completing the task, a team member will have to move the task to the **Testing section** for any verification and validation process.

After the respective champion has tested the task, team members will have to move it to **the To be confirmed section** for every stakeholder to review. If the task succeeds in the review, it will then be transferred to the **Done section.** Done section will denotes as the end of the life for task.



8.2.4. Issue Categories

Below is the classification for issue categories in development of Document submission system:

High-level issue:

1. Work breakdown structure creation failure (not enough detailed)
2. Module integration failure
3. Ambiguous system requirement
4. Additional system requirement
5. Review test failure
6. Quality review test failure

Low-level issue:

1. Functional errors. (Certain button did not work properly)
2. Syntax errors
3. Logic errors (infinite loop)
4. Calculation Errors

Standard issue:

1. Team communication issue.
2. Client communication issue
3. Unrealistic Schedule and project timeline

9. Chapter 9 – Tools and Methodologies

9.1. Tools

9.1.1. Microsoft Word

The written documentation will be in Microsoft Word.

9.1.2. Microsoft Project

Enables our team to keep track of the progress made to various tasks. This will help manage our workloads as we develop a schedule and timeframe for when a task is to be completed as well as allocating the various tasks to each team member and tracking the progression of the task.

9.1.3. Trello Board

Trello will help track Team 28 on their overall tasks.

9.1.4. Amazon RDS

A free relational database for MySQL.

9.1.5. Amazon EC2

It provides scalability as our team can focus on the development and deployment of the application faster and offers the ability to scale upwards or downwards based on the requirements and the demand.

9.1.6. OneDrive

OneDrive will be used to store all the documents and for the team to access immediately.

9.1.7. Visual Studio

Team 28 will be using Visual Studios to code for the Document Submission System.

9.1.8. GitHub

The source code written for this project will be uploaded to GitHub for storage.

9.1.9. Discord/Microsoft Team

Team 28 will be using Discord for group meetings every Thursday and Microsoft Team for meetings with both the supervisor and the client on Fridays. It is expected that all members have microphones to speak.

9.2. Design Methodology

Our team will adopt the Agile SDLC development method with scrum model. This enables us to focus on making decisions. Agile SDLC with scrum model helps our team to produce the application on time and effectively as it provides:

- A framework outlining the tasks to be completed
- Helps planning, scheduling and estimation
- Functionalities can be development and demonstrated quickly
- Allows to keep track of the project
- Lower production costs

- Increases development speed
- Offers flexibility
- Iterative testing
- More interaction with the client

Agile SDLC with scrum method is made up of these stages which are:

1. **Product Backlog Creation**

   Outlines the list of features that will be implemented during the development phase. It is set in order of priority and each item feature has a "User story" and each has a different ID.

2. **Sprint Planning**

   Depending on the workload required to finish a task, it must be determined how long a sprint will last as having short sprints will result in more frequent feedback where bugs, errors and issues can be identified early and resolved in due time.

3. **Working on Sprint/Scrum Meetings**

   Over the course of the project, we will be using a variety of tools to keep track of process during the duration of the sprints. These tools include Microsoft Project and Trello Board where our team can keep track of what is completed and what needs to be completed by our target date.

   To ensure that we know where we are at, we will be using Microsoft Team and Discord to hold scrum meetings to inform each other of our progress on the tasks that we are allocated to and have a set of objectives we need to complete during the sprint. Also, each scrum meeting, a team member will be allocated the role of scribe to write down the details of the meeting.

4. **Testing/Sprint Review**

   The result of each sprint potentially has a workable product which is shown to the client for feedback. The team will make changes or adjustments based on the client's review for the next sprint.

5. **Retrospective**

   The team will discuss the results and determine how to improve the development process for the next phase. Using Microsoft Team and Discord will allow us to interact with each other and discuss what went well, what improves we need to make and how we are going to make those improvements so we as a team can planning for the next sprint.

10. Chapter 10 – Records collection, maintenance, retention

Any descriptions, agendas, minutes undertaken from meetings with client are included in the project team's Github repository as detailed under subversion procedures. Any notes or minutes will be subjected to approval by all members of the team prior to inclusion in repository and all documents will be readily available in the repository for the course of the project.

11. Chapter 11 – Risk Management

11.1. Purpose

Risk Management is essential in a project to identify the weaknesses and strengths during the completion of the project. It helps to deliver the project on time with high quality.

11.2. Categorization

The risks identified for this project can be categorized as follows.

- Risks regarding management
- Risks regarding work to be done
- Risks regarding the client

The risks identified in these categories are stated in the following sections. For each risk, a description, the probability of occurring, the impact of it on the project and the actions that are taken to prevent or reduce them are given.

The probability of a risk occurring and the impact of a risk if it does occur can be identified as being low, moderate or high. The actions taken are categorized as preventative and reductive, preventative actions aim to lower the chances of risks occurring and reductive actions reduce the effects of risks if they arise.

11.3. Risks regarding work to be done

Corruption of the project repository

– Probability: Low.

– Impact: High resulting in loss of project work.

– Reductive Action: Updating the backups weekly can reduce the impact.

Poor quality code

– Probability: Moderate.

– Impact: High, poor-quality code increases the time taken to produce the output. The produced outcome will not be valid to client requirements.

– Preventative Action: Carrying out code reviews. Creating clear code standards and guidelines. Testing all the codes for bugs.

Team member leaving the project

-Probability: Low

-Impact: High, A team member may leave the project during the project completion period which may delay the project outcomes.

– Preventative Action: sharing key information with the group and the supervisor during the weekly stand-up meetings. Working in pairs, common code ownership.

Shortage of time

-Probability: Moderate

-Impact: Moderate, it might be hard to estimate the amount of work to be done in a given period.

-Reductive: Involving the team members in planning and estimating the project outcomes. Getting frequent feedback from the client and discussing the products regularly.

11.4. Risks regarding management

Illness or sudden absence of team leader

-Probability: Low

-Impact: High, A team leader is essential to plan and guide the team.

-Reductive: Appoint a temporary team leader.

SQAP not appropriate for project purposes

– Probability: Low.

– Impact: High, failure to follow SQAP would reduce the quality of the product.

– Preventative Action: Involving the team members in the discussion regarding the production of the SQAP and check the outcome with project supervisor.

11.5. Risks regarding the client

Scope Variation

– Probability: Moderate.

– Impact: Moderate, the workload for the period increases which will result in issues in the timeline.

– Preventative Action: discussion of the requirements for the project and the official SRS document early in project.

Client unavailable

– Probability: Moderate.

– Impact: Low, Client may be unavailable for questions raised during the project.

– Reductive Action: Communicate the key questions and prominent features of the project with the client before they become critical.

Client abandons project

– Probability: Low.

– Impact: High, there will be no more work in the project to complete.

– Reductive Action: Ask the client for as many modules as possible and requirements.

12. Chapter 12 – Appendix