# **Quality Assurance Plan**

This section is not limited to (apply what you have learnt in FIT2107):

* Who will be responsible for this process
* Extent of software testing
* **Definition of Done**

A task or feature is considered "completed" only if it meets the following criteria:

* Successfully completes all testing by passing all the test cases.
* Verify that the product feature adheres to the requirements and complies with ethical guidelines.
* Code has been documented and reviewed by each member, and feedback on each part of the code has been addressed.
* Sufficient test cases have been created, and the code successfully passes all test cases while maintaining good performance.
* The code merges with previous code seamlessly, without introducing new errors.
* The application's interface provides a pleasant viewing experience for the user.
* Code artifacts
* CI/CD software and tools, including advantages and disadvantages.
* Proposed process flow (within internal agile team and specialist team)

(<= 350 words)

**Non Functional Requirements**

Non functional requirements have a great impact on the system, even though they are not specified by the clients they are critical to client satisfaction. Below the following nonfunctional requirements are discussed: maintainability, portability, security, accessibility, and overall usability.

## Maintainability

How will you ensure that the software is easily maintainable? [~100-150 words]

We will ensure that the software is easily maintainable by implementing a modular design that isolates different functionalities into separate, independent components. This modularity allows for easier updates and modifications without impacting the entire system. Consistent coding standards will be enforced across the development team to maintain code quality and readability. Regular code reviews will be conducted to identify and address issues early, reducing technical debt and preventing code rot. Additionally, comprehensive documentation will be maintained, detailing the system's architecture and key processes, making it easier for any developers working with any task to understand and work with the codebase in the future.

## Portability

How is your delivery going to be portable? [~100-150 words]

## Security

How will your system manage user data, ensure secure interactions? [~100-150 words]

## Overall usability

The five main characteristics of usability: effectiveness, efficiency, engagingness, error tolerance and ease of learning, will be incorporated into our software. These can all be measured and tested for using industrial standard metrics. They are documented below:

[Have at least 1 test for each usability characteristic. If not applicable, argue why you cannot test the characteristic]

| **Name** | **Scale** | **Meter** | **Target** | **Constraint** |
| --- | --- | --- | --- | --- |
| *Effectiveness* | Completion rate of tasks | Test users that are new to the system are given tasks(eg: to create a project) to complete in the system , the number of tasks successfully completed can be recorded as a  percentage of the total tasks performed. | Target of > 80% completion rate | Completion rate should not fall < 60% |
| *Effectiveness* | Number of errors | Test users are given a set of tasks, the number of errors are recorded as a percentage. | Error rate < 20%. | Error rate should never fall < 50% |
| *Efficiency* | Task completion Time | ... |  |  |
|  | Response time | Time taken for operations |  |  |
| *Engagingness* |  |  |  |  |
| *etc..* |  |  |  |  |

## Accessibility

How is your design going to account for disabilities? [~100-150 words]

# **Git Repository and Branching Model**

Our team will utilise Git as the primary version control system, ensuring seamless collaboration and version tracking. The repository will be structured to support efficient branching and merging processes. The main branch will always contain production-ready code, while feature branches will be used for development tasks. Regular backups and monitoring will be conducted to maintain the integrity of the repository.

# **Commits**

Describe your team’s commit guidelines and rules. It includes frequency, message style, tags, milestone etc. (<=250 words)

Commit guidelines are designed to maintain a clean and organised commit history. Team members are expected to commit frequently, ensuring each commit addresses a single, logical change. Commit messages will follow the format "task : description", providing clarity and context for each change. Tags will be used to mark milestones. All commits should be linked to corresponding tasks or issues, ensuring alignment with project goals.

**Pre-Merge Procedures**

Before any code is merged into the main branch, it must pass all relevant test cases, ensuring that the codebase remains stable and functional, minimising the risk of regression. Team members are encouraged to promote collaboration and conduct pre-merge validations with the main code. This involves running tests in a staging environment and reviewing potential integration issues early.

**Code Review**

Code quality and standards compliance will be maintained through diligent code reviews. Each piece of code will be reviewed by at least one other team member before it is merged. This process not only ensures code integrity but also promotes knowledge exchange and collaboration among team members. Constructive feedback will be provided to help improve the overall quality of the codebase.

# **Branch**

Not limited to:

* Locked branch and restrictions
* Branch naming conventions
* Branches depth level rules

(<= 350 words)

# **Merge Requests**

Not limited to:

* Who will approve the request
* Testing with CI/CD
* Giving labels/tags/milestones

(<= 350 words)

# **Rules to Follow**

Summarise the description of rules above in the table below. You can use this as the reference to raise concerning problems.

| **Code** | **Subject** | **Description** | **If violated** |
| --- | --- | --- | --- |
| M1.1 | Merge Requests | Delete the source branch after merging. | RTE contacts that team member to delete the branch. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |