Version 2026.03

TinQwise Policy

014.1

Principles of Engineering

Validity and document management

Version	Date	Author	Description
01	2023-11-09	Okke Formsma	Document creation
02	2024-12-12	Emil de Valk	Annual review & update
03	2025-11-28	Emil de Valk	Annual review & update

This document is valid as of January 1st, 2026.

The owners of this document are the C.E.O and the Head of Engineering, who must check and, if necessary, update the document at least once a year.

TinQwise reserves the right to update this Policy to reflect changes in our practices and regulatory requirements. Data subjects will be informed of material changes. Previous versions of this procedure will be stored for a period of 5 years, unless specified otherwise by legal or contractual requirement.

Table of Contents

tinQwise

1. Relevant to

Management	Finance / Legal	HR	Office Management	Marketing, Sales and Ac-count Management	Professional services	IT Support	Product & Engineering	DevOps	Support & Maintenance
X							X	X	X

2. Purpose

The purpose of this document is to define guidelines for engineering of TinQwise Platform software and systems.

3. Scope

At TinQwise, we want to ensure that our software developers are aligned with our core principles to drive the development of our SAAS solutions. These principles provide a clear direction for our engineering team and serve as the foundation of our work.

4. Principles

There are 3 main areas of focus:
1. security and performance;
2. simplicity;
3. and iteration speed.

These principles go hand in hand to create a high-quality experience for our customers.

4.1 Security and Performance

Prioritize Security: Security is non-negotiable. We're responsible for safeguarding our customers' data, and it's our duty to ensure their trust isn't compromised. Our engineering practices include robust encryption, strict authentication, and effective access control measures.

Performance Matters: We know that responsive SAAS applications are essential. We're dedicated to optimizing performance across the board. Constant performance monitoring and improvements are integral to our work to ensure our software remains swift and can handle increased workloads.

Scalability is Key: As our customers grow, so must our solutions. Scalability is woven into our engineering fabric, enabling smooth expansion without sacrificing security or performance.

Resilient Data Management: Our business revolves around data. Redundant data storage and effective backup strategies are embedded in our systems to guarantee data integrity and accessibility under any circumstances.

4.2 Simplicity

User-Focused Design: Our design philosophy revolves around user-friendliness. We create interfaces that are intuitive and user-friendly, ensuring that users can efficiently complete tasks without confusion.

Clean, Understandable Code: We follow a "Keep It Simple" mantra when writing code. Clear, maintainable code helps reduce bugs, accelerates development, and eases collaboration.

Minimal Dependencies: We keep external dependencies to a minimum, reducing complexity and ensuring that our software relies on solid, well-documented libraries and frameworks.

4.3 Iteration Speed

Agile Approach: We're committed to an agile development approach that encourages rapid iterations, continuous improvement, and adaptable planning. Teams have the authority to make decisions, prioritize tasks, and deliver value swiftly.

Rapid Prototyping: Rapid prototyping helps us quickly explore new concepts and gather early feedback from stakeholders. This approach allows us to identify issues early, learn from our mistakes, and iterate towards better solutions.

Automated Testing: Our emphasis on automated testing provides rapid feedback and ensures software quality. It instils confidence in releasing new features and updates while maintaining the stability of existing functionality.

In summary, these principles - security and performance, simplicity, and iteration speed - are the foundation of our engineering work at TinQwise. By adhering to these principles, we aim to empower our software developers to consistently deliver high-quality SAAS solutions that meet and exceed the expectations of our customers. Through continuous improvement and a relentless focus on customer value, we remain at the forefront of SAAS innovation.

5. Key processes

There are a few key processes that help us spread the principles throughout the team and make sure the first two principles (security and performance, and simplicity) are achieved without sacrificing the third (iteration speed). These processes are the code review process and the release process.

5.1 Code review process

We believe in the importance of maintaining the highest quality in our applications. To achieve this, we have established a code review process as a standard practice. However, we also understand that there are situations where code reviews can slow down the development process.

Categorizing Changes

To strike the right balance, we categorize changes into two groups: 'normal' and 'trivial.' It is the responsibility of our developers to determine which category a change falls into. These categories are defined as follows:

Normal Changes:
- Affect more than three files or involve more than ten lines of code.
- May have performance or security implications.

Trivial Changes:
- Typically include minor styling adjustments in the frontend.
- Involve adding fields to APIs.
- Consist of changes in fewer than three files and less than ten lines of code.

It's important to note that 'trivial' changes do not encompass any alterations that might pose security risks. If a developer has any doubts about potential security or performance implications, the change cannot be classified as 'trivial.'

Developer Responsibility
The responsibility for categorizing a change as 'trivial' or 'normal' rests with our developers. A 'trivial' change can be merged into the 'development' Git branch immediately for inclusion in the next release. However, if the 'trivial' change is a hotfix, it must first be deployed to the acceptance (QA) environment and tested before it is released to production.

Regular Review and Improvement
As part of our commitment to continuous improvement, we conduct a two-weekly retrospective where all commits from the previous sprint are collectively reviewed and evaluated by the team. This process ensures that we maintain a high standard of code quality and encourages ongoing learning and growth within our development team.

5.2 Release process

All code must run through the Continuous Integration/Continuous Deployment pipeline with automated tests before it can be released to development, QA or production.

At the end of every (two-week) sprint, there is a formal release. This release contains all new functionality that has been developed during the sprint. The timeline is as follows:

- Two days before the release:

tinQwise

    - (Developer) refresh of acceptance environment by creating a new copy of the production environment
    - (Developer) creation of release branch. Release to acceptance environment.
    - (Developers, testers) test existing and new functionality
- One day before the release:
    - (Developers, testers) test existing and new functionality
    - (Developers) bugfixes
- One hour before release:
    - (Head of Engineering, Product Owner): go/no go decision on the release
- Release:
    - (Head of Engineering) deploys release to production
- After release:
- (Head of Engineering) verifies success of deployment. In case of problems, initiates a rollback to the previous version.

If the Head of Engineering or Product Owner are not available for the go/no-go meeting, they must delegate this responsibility to a suitable replacement.