ISO/IEC/IEEE 12207:2026 Systems and software engineering, Software life cycle processes (2nd edition, April 2026)

Process framework for the software life cycle

63 requirements. There is no accredited certificate for this standard. As of 07/2026.

ISO/IEC/IEEE 12207 defines a framework of 30 processes covering the complete software life cycle, from agreement through development and operation to disposal. There is no accredited certification scheme: the standard explicitly treats conformance as a self declaration in three variants (full conformance to outcomes, full conformance to tasks, or tailored conformance with a documented tailoring record), so the result of an assessment is a conformance statement and not a certificate. Organizations that need an external judgement commission a process assessment based on the ISO/IEC 33000 series, which delivers a capability profile in the form of a report.

Company	Date	Prepared by

6.1 Agreement Processes 6

6.1.1 The Acquisition process is in place: the need for externally sourced software or services is identified, suppliers are selected, and an agreement is concluded and monitored.

Document

What proves it: Evidence includes tender or requirement documents, comparisons of offers, signed contracts or purchase orders, and acceptance records. In small teams a short acquisition ticket per purchase in the issue tracker, covering the need, the comparison of alternatives, and the approval, is sufficient.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.1.1.2 The outcomes of acquisition are achieved: requirements for the acquired item are defined, the agreement is concluded, and fulfilment is confirmed.

What proves it: Acceptance criteria in the contract or in the ticket, plus the acceptance test or acceptance record. For open source dependencies, the lockfile with a documented selection criterion and the license scan serve as acceptance evidence.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.1.1.3 The acquisition activities are carried out: identification of the need, supplier selection, conclusion of the agreement, progress monitoring, and acceptance are performed in a traceable way.

What proves it: Records from the issue tracker and the contract folder, supplemented by supplier status reports. A supplier list with the date of the last evaluation is enough for small organizations.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.1.2 The Supply process is in place: the organization reviews requests, submits an offer, concludes an agreement, delivers the product or service, and formally closes the agreement.

Document

What proves it: Offers, order confirmations, delivery and release notes, handover records, and invoices. Release notes per version are the most practical delivery evidence for software teams.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.1.2.2 The outcomes of supply are achieved: the scope of delivery is agreed, the product is handed over, and responsibility for it is settled.

What proves it: Signed release artifacts with version number, changelog, and a handover message to the customer. In addition, support and warranty arrangements in the contract terms.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.1.2.3 **The supply activities are carried out: feasibility is assessed, the delivery is executed, and the agreement is closed once it has been fulfilled.**

What proves it: A record of the feasibility or effort estimate in the offer, the deployment log of the CI pipeline as delivery evidence, and a short closure record per order.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.2 Organizational Project-Enabling Processes 9

6.2.1 **Life Cycle Model Management is in place: the organization defines, maintains, and makes available the life cycle models, processes, and procedures that projects use.** Document

What proves it: A lean process description in the repository, for example CONTRIBUTING.md plus a one page development workflow with branching model, definition of done, and release cadence. Changes to it are datable through the Git history.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.2.1.2 **The outcomes are achieved: the life cycle models in use are defined, projects are aware of them, and they are reviewed regularly for effectiveness.**

What proves it: Retrospective notes or an annual review entry in the repository that gives the reason for adapting the workflow. Pull requests against the process documents are the evidence of effectiveness.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.2.2 **Infrastructure Management is in place: the infrastructure needed for development, test, and operation is defined, provided, and maintained.**

What proves it: Infrastructure as code in the repository (Terraform, Docker Compose, CI runner configuration), plus a list of the services in use with their owners. For small teams a single page with environments, access, and backup rules is enough.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.2.2.3 **The infrastructure activities are carried out: needs are identified, resources are acquired, operated, and retired when no longer required.**

What proves it: Change history of the infrastructure configuration in Git, monitoring dashboards, and tickets for environment changes. Decommissioned environments are documented in the ticket with a date.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.2.3 **Portfolio Management is in place: projects are initiated, prioritized, resourced, evaluated on an ongoing basis, and terminated when they no longer deliver value.**

What proves it: A product roadmap or portfolio board with priorities and decision dates, plus records of initiatives that were stopped. A roadmap document with a quarterly review is enough for small organizations.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.2.4 **Human Resource Management is in place: the required competencies are determined, staff are qualified, and knowledge is built up and retained.**

What proves it: A competence matrix per role, training and certificate records, an onboarding checklist. In small teams a skills table showing gaps and planned training, together with evidence of completed training, is sufficient.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.2.5 **Quality Management is in place: quality objectives, the quality policy, and responsibilities are defined, and achievement of the objectives is evaluated.** Document

What proves it: A short quality policy in the repository, defined quality objectives (test coverage, defect density, response time for incidents), and the corresponding metrics from CI and the issue tracker.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.2.5.2 **The outcomes of quality management are achieved: quality objectives are measurable, deviations are detected, and corrective actions are tracked.**

What proves it: Trend reports from the CI pipeline (coverage, failed builds), defect statistics from the issue tracker, and tickets for the resulting actions with a completion date.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.2.6 **Knowledge Management is in place: knowledge, skills, and reusable assets are captured, shared, and kept available for later projects.**

What proves it: An internal knowledge base or wiki, architecture decision records in the repository, reusable libraries with documentation. Postmortem notes after incidents are strong evidence.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.3 Technical Management Processes 16

6.3.1 **Project Planning is in place: scope, deliverables, schedule, effort, roles, and interfaces are planned, and the plan is kept up to date.** Document

What proves it: A project plan or backlog with milestones, estimates, and owners. A maintained sprint or milestone board in the issue tracker meets the requirement as long as scope and dates are visible.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.3.1.2 **The planning outcomes are achieved: tasks and resources are allocated, roles are named, and the plans are agreed with those involved.**

What proves it: Assigned issues with an owner and a target version, plus the milestone overview. Capacity assumptions are noted briefly in the planning ticket.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.3.1.3 **The planning activities are carried out: the project scope is defined, the plans are produced, and they are updated whenever changes occur.**

What proves it: Version history of the plan or roadmap document in Git, and change comments on milestones. This makes every update datable.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.3.2 **Project Assessment and Control is in place: progress against plan is assessed, deviations are detected, and control actions are taken.** Document

What proves it: Status reports, sprint reviews, burndown or throughput metrics from the issue tracker, and records of the decisions taken when deviations occurred. A short weekly status note is sufficient.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.3.3 **Decision Management is in place: decisions are prepared in a structured way, alternatives are evaluated, and the selection is justified and recorded.** Document

What proves it: Architecture decision records in the repository, covering context, options, decision, and consequences. For smaller decisions a reasoned comment in the pull request is enough.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.3.3.2 **The outcomes of decision management are achieved: the need for a decision is recognized, alternatives are evaluated, and the decision taken is traceable together with its rationale.**

What proves it: The ADR folder with sequential numbering and status (proposed, accepted, superseded). The Git history evidences the date and the author.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.3.4 **Risk Management is in place: risks are identified, analyzed, treated, and monitored on an ongoing basis.** Document

What proves it: A risk register with likelihood, impact, treatment, and owner. Technical risks can be tracked as issues with a risk label, together with the results of dependency and vulnerability scans.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.3.4.2 **The outcomes of risk management are achieved: risks are prioritized, thresholds for action are defined, and the effectiveness of the treatment is tracked.**

What proves it: Update history of the risk register, closed risk tickets with a rationale, and reports from automated dependency scanning showing the remediation date.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.3.5 **Configuration Management is in place: configuration items are identified, baselines are defined, changes are controlled, and the status of every item can be determined at any time.** Document

What proves it: Git as version control with tags for releases, protected main branches, lockfiles for dependencies, and a software bill of materials (SBOM) per release. The tag history is the baseline record.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.3.5.2 **The outcomes of configuration management are achieved: baselines are established, changes are traceable, and releases are reproducible.**

What proves it: A reproducible build from a Git tag through the CI pipeline, plus build logs and an artifact registry with checksums.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.3.5.3 **The configuration management activities are carried out: change requests are evaluated, approved or rejected, and their incorporation is verified.**

What proves it: Pull request history with mandatory review and a linked issue, and merge rules in the repository (branch protection) as evidence of approval control.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.3.6 **Information Management is in place: the information to be created and retained is determined, and it is maintained, protected, and made available to those entitled to it.** Document

What proves it: A records concept covering storage location, owner, retention period, and access rights. Repository permissions and backup logs evidence protection and availability.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.3.7 **The Measurement process is in place: information needs are determined, suitable measures are defined, collected, analyzed, and used for decisions.**

What proves it: Metrics from CI and the issue tracker, such as lead time, defect rate, test coverage, deployment frequency. What matters is the evidenced path from the measure to the decision, for example in the retrospective record.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.3.7.2 **The outcomes of measurement are achieved: the collected data are valid, the analysis is documented, and the results are communicated.**

What proves it: Automatically generated pipeline reports with a timestamp, and a regular metrics summary posted in the team channel or wiki.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.3.8 **Quality Assurance is in place: products and processes are checked independently against the defined criteria, and deviations are recorded and resolved.** Document

What proves it: Mandatory four eyes review in the pull request, linter and test runs as a merge gate, and recording of the deviations found as issues. A small team achieves independence through rotating reviewers rather than a dedicated QA department.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.3.8.2 **The outcomes of quality assurance are achieved: process conformance and product conformance are evaluated, and corrective actions are tracked to closure.**

What proves it: Review comments with a resolution status, closed bug tickets referencing the fix commit, and a green pipeline run for the release tag.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4 Technical Processes 26

6.4.1 **Business or Mission Analysis is performed: the problem to be solved, the solution space, and the constraints of the undertaking are determined.**

What proves it: A product vision, problem statement, or business case in the wiki, supplemented by market analysis or user interviews. An epic with a problem description and objective in the issue tracker is enough for small undertakings.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.2 **Stakeholder Needs and Requirements Definition is performed: stakeholders are identified, their needs are elicited, and they are transformed into verifiable stakeholder requirements.** Document

What proves it: A stakeholder list, interview notes, user stories with acceptance criteria. The acceptance criteria in the ticket are the verifiable form of the requirement.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.2.2 The outcomes are achieved: the stakeholder requirements are agreed, prioritized, and traceable back to the underlying need.

What proves it: A prioritized backlog that links story to epic, plus the customer approval comment in the ticket.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.3 System Requirements Definition is performed: the technical requirements for the software system are derived from the stakeholder requirements, including quality characteristics and constraints. Document

What proves it: A requirements document or structured issues covering functional and non functional requirements (performance, security, accessibility). Linking each requirement to a test case makes verifiability explicit.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.3.2 The outcomes of requirements definition are achieved: the requirements are unambiguous, free of contradictions, verifiable, and traceable back to the stakeholder requirements.

What proves it: A traceability matrix, or alternatively issue links from story to test and to commit. Automated acceptance tests are at the same time the evidence of verifiability.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.4 System Architecture Definition is performed: an architecture with views, building blocks, and interfaces is defined and evaluated against the requirements. Document

What proves it: An architecture overview in the repository (for example arc42 or C4 diagrams as an image plus text), together with architecture decision records holding the evaluated alternatives.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.4.2 The outcomes of system architecture definition are achieved: the architecture is evaluated against the requirements, alternatives are weighed, and the selection is justified.

What proves it: An ADR weighing options and consequences, plus the results of an architecture review in the pull request or in a meeting record.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.5 Design Definition is performed: designs exist for the architectural building blocks that are sufficient for implementation, including the data model and interface contracts.

What proves it: Interface specifications (OpenAPI, GraphQL schema, type definitions), database migration scripts, and design notes in the pull request. The code itself, with its types and module boundaries, is acceptable design evidence provided the decisions are documented.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.5.2 The outcomes of design definition are achieved: the design characteristics are traceable to the architecture and feasibility is confirmed.

What proves it: References from the pull request to the ADR and to the requirement, and prototypes or spikes with a result note as evidence of feasibility.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.6 **System Analysis is performed: technical questions are investigated with analytical methods in order to substantiate decisions on architecture, design, or risks.**

What proves it: Load measurements and benchmark results, cost or trade off analyses, threat model notes. Results belong with the related ADR as a short analysis note.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.7 **Implementation is performed: the software elements are realized in line with the design and the coding rules, and the results are placed under configuration control.**

What proves it: Git history with commits that reference issues, linter and formatting rules as a merge gate, and successful builds in the CI pipeline.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.7.3 **The implementation activities are carried out: coding standards are defined, the realization is reviewed, and the elements are made available for integration.**

What proves it: Linter and formatter configuration files in the repository, passed pull request reviews, and published build artifacts in the registry.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.8 **Integration is performed: software elements are progressively assembled into a working system and the interfaces are checked.**

What proves it: Continuous integration runs on every merge, integration tests and contract tests against the interfaces, plus the build logs with timestamps.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.8.2 **The outcomes of integration are achieved: the integration strategy is defined, the interfaces are consistent, and integration defects are recorded and resolved.**

What proves it: The pipeline definition in the repository as the integration strategy, failing runs followed by green runs as evidence of defect resolution, and bug tickets with the fix commit.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.9 **Verification is performed: for every specified requirement there is verifiable, traceable evidence that the software as built actually satisfies it.** Document

What proves it: Automated unit, integration, and regression tests with a test report from the CI pipeline, static analysis, and code reviews. The test report per release tag is the central piece of evidence.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.9.2 **The outcomes of verification are achieved: verification criteria are defined, the checks are executed, and deviations are recorded.**

What proves it: Defined acceptance thresholds (for example a minimum test coverage, no critical findings), archived test reports, and an issue for every deviation.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.9.3 **The verification activities are carried out: the scope and the methods of verification are planned, executed, and the results are analyzed.**

What proves it: A test plan or test strategy as a short page in the repository, the pipeline configuration with its verification stages, and the analysis in the release approval.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.10 **The Transition process is performed: the system is installed in its target environment, handed over ready for use, and its operational capability is confirmed.** Document

What proves it: A deployment runbook, deployment logs from the pipeline, smoke tests after the rollout, and a documented rollback plan. Release notes evidence the handover to users and to operations.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.11 **Validation is performed: under the intended conditions of use it is demonstrated that the software actually meets the original needs of the stakeholders.** Document

What proves it: Acceptance tests with the customer, a beta or pilot phase with user feedback, and acceptance criteria in the ticket marked as met. Usage data from operation gives the validation additional support.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.11.2 **The outcomes of validation are achieved: validation criteria are defined, validation takes place in a representative environment, and deviations are recorded.**

What proves it: An acceptance record with date and participants, results from the staging or pilot environment, and feedback tickets with their resolution status.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.12 **The Operation process is in place: the system is operated as intended, it is monitored, and users are supported.**

What proves it: Monitoring and alerting with defined thresholds, incident tickets with response times, and a support mailbox or ticket channel. An operations manual of a few pages is enough for small organizations.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.12.2 **The outcomes of operation are achieved: the operation strategy and its measures are defined, incidents are recorded, and performance is evaluated against expectations.**

What proves it: Availability and error metrics from monitoring, incident logs, and postmortems with the actions derived from them.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.13 **The Maintenance process is in place: defect fixes, adaptations, and improvements are recorded, evaluated, implemented, and released without compromising the integrity of the system.** Document

What proves it: Bug and change tickets with priority and root cause, hotfix branches, regression tests before release, and release notes per maintenance version. Regular dependency updates are evidenced through the lockfile history.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.13.2 **The outcomes of maintenance are achieved: changes are analyzed, their impact is assessed, and the effectiveness of the fix is confirmed.**

What proves it: An impact analysis in the ticket, a regression test run after the fix, and confirmation by the reporter. Security updates are evidenced through vulnerability scan reports showing the remediation date.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.14 **The Disposal process is in place: the retirement of systems or parts of systems is planned, data are migrated or deleted, and affected parties are informed.** Document

What proves it: A retirement plan with dates, evidence of data migration or deletion (including backups), a shutdown record for the infrastructure, and the announcement to users or customers.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.14.2 **The outcomes of disposal are achieved: retention obligations are met, environments are shut down, and responsibility for the system is ended.**

What proves it: An archive of the data subject to retention obligations with the retention period stated, a ticket for shutting down resources with a date, and a deletion confirmation from the hosting provider.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

A Conformance and Tailoring 6

4.1 **The scope of application is defined (clause 4.1, General): it is determined whether the standard is applied to a project, an organization, or an agreement, and which processes are covered.** Document

What proves it: A short scope statement in the process documentation naming the project, the product, and the process groups applied. One page in the repository is enough.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

4.2 **The chosen conformance variant is declared: either full conformance to all outcomes of the claimed processes, or full conformance to all tasks of the claimed processes.** Document

What proves it: A conformance statement as a separate document that names the variant and points to the evidence for each process. This produces a self declaration, not a certificate.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

4.3 **Where tailored conformance is claimed, the tailoring is documented: it is stated which processes, tasks, or outcomes have been omitted, modified, or added.** Document

What proves it: A tailoring table with the columns process, decision (adopted, adapted, omitted), rationale, and risk. For small teams this document is the decisive piece of evidence.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

5 **The key concepts are applied to the organization: the relationship between the life cycle model, the project structure, and the processes in use is presented in an understandable way.**

What proves it: An overview page mapping the chosen approach (for example iterative, continuous delivery) onto the 30 processes. A diagram plus a table is enough.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

A.1 **Tailoring is handled as a deliberate decision: tailoring is permitted, but it is only applied where the constraints of the undertaking justify it, and it is always justified.** Document

What proves it: A rationale column in the tailoring table referring to project size, risk class, contractual situation, or legal requirements. Blanket removals without a rationale are not valid tailoring.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

A.2 **The tailoring process is carried out: influencing factors are identified, stakeholders are involved, and tailoring decisions are taken and documented.** Document

What proves it: A record or ticket for the tailoring decision naming the roles involved, plus the versioned tailoring table in Git. Effects on risks are noted in the risk register.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

ISO/IEC 25010:2023 Systems and software engineering, SQuaRE, Product quality model

Quality model for software products

40 requirements. There is no accredited certificate for this standard. As of 07/2026.

ISO/IEC 25010:2023 is a quality model, not a requirements standard with auditable clauses, and therefore cannot be certified under accreditation. The model serves as a grid for stating quality requirements completely; measurement follows ISO/IEC 25020 and evaluation follows ISO/IEC 25040, and the result is a traceable evaluation report, not a certificate. Advertising a supposed ISO 25010 certification is not permissible.

Company	Date	Prepared by

1 Functional Suitability 3

1.1 The set of functions covers all specified tasks and user goals the product is used for (Functional completeness). Gaps between documented requirements and implemented functions are known and assessed.

What proves it: Evidence comes from a traceability matrix linking requirement to test case, measurable as requirements coverage in percent. In small teams an issue tracker with one label per requirement plus one acceptance test per user goal is enough.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

1.2 For the defined inputs the product delivers results with the required degree of precision (Functional correctness). Tolerances and permissible deviations are specified.

What proves it: The measurable figure is defect density per thousand lines of code or per function, backed by unit and integration tests comparing expected against actual results, plus bug statistics from the tracker. For calculation logic, keep reference data sets with expected results on file.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

1.3 The functions provided support the users' tasks without unnecessary steps and fit the intended use (Functional appropriateness).

What proves it: Evidence comes from task success rate and number of steps per user task in a small usability test with five participants, logged per task. Usage data or support requests that reveal detours round out the picture.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

2 Performance Efficiency 3

2.1 Response, processing and throughput times stay below defined thresholds (Time behaviour). The thresholds are defined for the typical usage scenarios.

What proves it: Measurable through load time and interaction latency captured with Lighthouse or Web Vitals (LCP, INP), plus server response times from the logs at the 95th percentile. Record the per scenario thresholds in a short performance budget.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

2.2 Consumption of processor time, memory, storage and network bandwidth stays within the defined limits (Resource utilization).

What proves it: Evidence comes from monitoring values for CPU, RAM and data volume under load, plus the shipped bundle size from the build report. For small businesses a measurement log from one load test against the limits of the target hardware is sufficient.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

2.3 **The maximum number of concurrent users, transactions or records is known and observed (Capacity).**

What proves it: Evidence comes from a load test (for example k6 or JMeter) with a documented saturation point and the point at which error rates start to rise. Summarise the result in a short capacity report stating the actual limit and the planned demand.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

3 Compatibility 2

3.1 **The product operates in a shared environment without impairing other products (Co-existence).**

What proves it: Evidence comes from a test run in the target environment with other software running in parallel, logging port conflicts, memory usage and file access. Containerisation with defined resource limits is the practical route for small teams.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

3.2 **Data exchange with other systems works through defined interfaces and formats (Interoperability).**

What proves it: Evidence comes from a versioned interface description (OpenAPI or schema file) and contract tests against that description. Verify import and export formats with real sample files and keep the results on file.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

4 Interaction Capability 8

4.1 **Users can tell without outside help whether the product fits their needs (Appropriateness recognizability). Purpose and benefit are evident from the interface and the entry level texts.**

What proves it: Evidence comes from a short first impression test: participants describe the purpose of the product after 30 seconds and the hit rate is recorded. Screenshots of the start page and the onboarding screens support the case.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

4.2 **New users reach the core tasks within a defined time and with a limited number of errors (Learnability).**

What proves it: Measurable as time to the first successful core task and failed attempts per task, captured in a test with five new users. Document onboarding, guided tour or demo data as supporting measures.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

4.3 **Operation is efficient and controllable, and recurring tasks can be carried out without detours (Operability).**

What proves it: Evidence comes from a click path analysis per standard task (target steps against actual steps) and from keyboard only operation without a mouse. The usability test log with the abandonment rate per task completes the record.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

4.4 **The product prevents operating errors or makes them reversible (User error protection). Critical actions require confirmation or can be undone.**

What proves it: Evidence comes from test cases with invalid inputs and boundary values, plus proof of validation, confirmation dialogue and undo function. Evaluate faulty inputs from the production log and track their frequency as a metric.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

4.5 **The interaction is designed to be appealing and keeps users motivated to stay with the task (User engagement).**

What proves it: Measurable through return rate, session duration and completion rate of started processes taken from privacy friendly usage statistics. A short satisfaction survey with a rating scale is enough on top of that, also for small businesses.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

4.6 **The product is usable by people with different abilities, languages and cultural backgrounds (Inclusivity).**

What proves it: Evidence comes from a check against WCAG 2.2 level AA: contrast ratios, keyboard navigation, screen reader labels, tested with axe or the Lighthouse accessibility audit plus a manual sample. Demonstrate language coverage and localisation through the completeness of the translation files.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

4.7 **Users receive help that matches the task when they need it, including in error situations and edge cases (User assistance).**

What proves it: Evidence comes from the manual, context sensitive help texts and understandable error messages that suggest a next action. Metric: the share of support requests that existing help could have resolved, taken from the ticket system.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

4.8 **The interface explains itself: states, the effects of actions and the inputs required are recognisable without prior knowledge (Self-descriptiveness).**

What proves it: Evidence comes from an interface review against a checklist: meaningful labels, visible system status, progress and loading indicators, units on numeric fields. An observation log from the user test adds weight to the record.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

5 Reliability 4

5.1 **In normal operation the product runs without failures caused by faults in the software (Faultlessness).**

What proves it: Measurable through defect density and mean time between failures (MTBF) from production logs, and through test coverage as a preventive metric. Error monitoring such as Sentry supplies the raw data with little effort.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

5.2 **The product is reachable and usable during the agreed time windows (Availability). The availability target is defined.**

What proves it: Evidence comes from uptime measurement by external monitoring, stated as availability in percent per month and compared with the agreed target. Record outages with duration and cause in a simple incident log.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

5.3 **In the presence of hardware or software faults the product stays in a defined, usable state (Fault tolerance).**

What proves it: Evidence comes from test cases with simulated failures (network down, database down, third party system not responding) and the documented behaviour, for example offline mode, retry or fallback display. File the results as a test log.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

5.4 **After a failure, data and operating state can be restored within the defined time (Recoverability).**

What proves it: The measurable figures are recovery time objective (RTO) and maximum tolerable data loss (RPO), evidenced by an actually performed restore test with date and duration. For small businesses a half yearly restore test with a short log is sufficient.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6 Security 6

6.1 **Data is accessible only to authorised people and systems (Confidentiality). Access rights are granted on a need to know basis.** Document

What proves it: Evidence comes from a role and permission concept, encryption in transit and at rest, and tests that confirm access is denied without authorisation. A header and TLS check of the application rounds this out.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.2 **Unauthorised changes to programs and data are prevented or detected (Integrity).**

What proves it: Evidence comes from checksums or signatures of the artefacts, input validation and a dependency scan (npm audit, Dependabot) showing that findings were remediated. Database constraints and migration logs further support this point.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.3 **Actions that were carried out cannot be denied afterwards (Non-repudiation).** Document

What proves it: Evidence comes from tamper resistant logs with timestamp and originator, and from signatures where the transaction has legal relevance. A central, write protected log target is enough for small businesses.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4 **Actions in the system can be traced unambiguously to a responsible person or service (Accountability).** Document

What proves it: Evidence comes from audit logs carrying a user identifier for every security relevant action, combined with a ban on shared accounts. Spot check: trace one transaction end to end in the log and document the result.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.5 **The identity of users and systems is verified reliably (Authenticity).**

What proves it: Evidence comes from the authentication method, the password policy and two factor authentication for administrative access, and from the validation of tokens and certificates. Test cases with expired and forged tokens demonstrate effectiveness.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.6 **The product withstands attacks and remains operational while under attack (Resistance).**

What proves it: Evidence comes from a vulnerability scan, static code analysis (SAST) and, where proportionate, a penetration test with a findings list and remediation status. For small businesses an automated scan in the pipeline plus a yearly short external test is the practical route.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

7 Maintainability 5

7.1 The product consists of separated components whose modification affects as few other components as possible (Modularity).

What proves it: Measurable through coupling and cohesion from a static analysis and through dependency graphs (madge, dependency-cruiser) showing that no cycles exist. Capture the module boundaries in a brief architecture sketch.

◯ Open ◯ In progress ◯ Met ◯ Not applicable

Evidence / justification

7.2 Components are designed so that they can be reused in more than one context (Reusability).

What proves it: Evidence comes from shared libraries or components with their own versioning and a documented interface. Metric: the share of reused components and the level of code duplication from the static analysis.

◯ Open ◯ In progress ◯ Met ◯ Not applicable

Evidence / justification

7.3 Causes of failures and the impact of planned changes can be determined efficiently (Analysability).

What proves it: Evidence comes from structured logging, error monitoring and readable code metrics such as cyclomatic complexity from SonarQube or ESLint. Metric: mean time to root cause from the ticket system.

◯ Open ◯ In progress ◯ Met ◯ Not applicable

Evidence / justification

7.4 Changes can be made without introducing new defects and without degrading quality (Modifiability).

What proves it: Evidence comes from regression tests in the pipeline and the change failure rate, meaning the share of defects introduced by changes. Mandatory code review and version control with traceable commits support this point.

◯ Open ◯ In progress ◯ Met ◯ Not applicable

Evidence / justification

7.5 Effective test criteria and tests can be established and executed for the product and its components (Testability).

What proves it: Measurable through test coverage in percent (lines and branches) from the coverage report and through the runtime of the test suite. An automated test run on every commit is the evidence that it works.

◯ Open ◯ In progress ◯ Met ◯ Not applicable

Evidence / justification

8 Flexibility 4

8.1 The product can be adapted to different or changing environments without having to be rebuilt (Adaptability).

What proves it: Evidence comes from configuration held outside the code (environment variables, configuration files) and from tests across several target environments, browsers or operating systems in a test matrix. File the result as a compatibility table.

◯ Open ◯ In progress ◯ Met ◯ Not applicable

Evidence / justification

8.2 The product handles growing or shrinking volumes of users and data without an unacceptable drop in performance (Scalability).

What proves it: Evidence comes from load tests with a rising number of users, tracking response time and error rate, complemented by the scaling strategy (horizontal or vertical). Metric: response time at single, double and fivefold load.

◯ Open ◯ In progress ◯ Met ◯ Not applicable

Evidence / justification

8.3 **The product can be installed, updated and removed reliably in the target environment (Installability).**

What proves it: Evidence comes from a reproducible installation guide and an automated installation test on a clean system, measured as success rate and time required. Run uninstall and rollback once as well and log the outcome.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

8.4 **The product can replace another product used for the same purpose while data and workflows are preserved (Replaceability).**

What proves it: Evidence comes from import and export functions in open formats and from a migration test with real legacy data, checked for completeness and data fidelity. Keep the data model and the migration scripts under version control.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

9 Safety 5

9.1 **The safe operating range is defined, and the product detects and handles any departure from that range (Operational constraint).**

What proves it: Evidence comes from documented operating limits (threshold values, permissible states) and tests that check the behaviour outside those limits. If the software carries no potential for harm to people or property, this point is typically not applicable; that classification must be justified and documented, not silently omitted.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

9.2 **Hazards that can arise from the product or its use are identified and assessed systematically (Risk identification).** Document

What proves it: Evidence comes from a hazard and risk assessment stating likelihood, severity and mitigating measure for each hazard, practical as a table in the quality folder. Not applicable only with a written justification that there is no potential for harm to people or property.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

9.3 **In the event of a malfunction the product transitions into a safe state (Fail safe).**

What proves it: Evidence comes from test cases with a forced fault and the documented transition into the safe state, for example shutting down, locking or falling back to a verified default value. For software without potential for harm this point is typically not applicable, but the justification still belongs in the evaluation.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

9.4 **The product warns of impending hazardous conditions early enough for a response to be possible (Hazard warning).**

What proves it: Evidence comes from an alarm and warning concept with thresholds, lead time and recipients, plus a test log of triggered warnings. Without potential for harm this point is typically not applicable; the justification has to be documented.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

9.5 **Interaction with other systems does not create new hazards (Safe integration).**

What proves it: Evidence comes from an interface risk review and integration tests with the connected systems, including the behaviour when a partner system fails or sends faulty data. For software without potential for harm this point is typically not applicable, and that has to be recorded with a justification.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification