TQS: Quality Assurance manual

**Bruno Moura (97151), João Teles (104360), Marcus Peterson (), Victor Melo ()**

v2023-05-19

[1 Project management 1](#_Toc132723568)

[1.1 Team and roles 1](#_Toc132723569)

[1.2 Agile backlog management and work assignment 1](#_Toc132723570)

[2 Code quality management 2](#_Toc132723571)

[2.1 Guidelines for contributors (coding style) 2](#_Toc132723572)

[2.2 Code quality metrics 2](#_Toc132723573)

[3 Continuous delivery pipeline (CI/CD) 2](#_Toc132723574)

[3.1 Development workflow 2](#_Toc132723575)

[3.2 CI/CD pipeline and tools 2](#_Toc132723576)

[3.3 System observability 2](#_Toc132723577)

[4 Software testing 2](#_Toc132723579)

[4.1 Overall strategy for testing 2](#_Toc132723580)

[4.2 Functional testing/acceptance 3](#_Toc132723581)

[4.3 Unit tests 3](#_Toc132723582)

[4.4 System and integration testing 3](#_Toc132723583)

# Project management

## Team and roles

**Team Leader** ➝ João Teles

Ensure that there is a fair distribution of tasks and that members work according to the plan. Actively promote the best collaboration in the team and take the initiative to address problems that may arise. Ensure that the requested project outcomes are delivered in time.

**QA Engineer ➝** Marcus Peterson

Responsible, in articulation with other roles, to promote the quality assurance practices and put in practice instruments to measure que quality of the deployment. Monitors that team follows agreed QA practices.

**DevOps Master ➝** Bruno Moura

Responsible for the (development and production) infrastructure and required configurations. Ensures that the development framework works properly. Leads the preparing the deployment machine(s)/containers, git repository, cloud infrastructure, databases operations, etc.

**Product Owner** **➝** Victor Melo

Represents the interests of the stakeholders. Has a deep understand of the product and the application domain; the team will turn to the Product Owner to clarify the questions about expected product features. Should be involved in accepting the solution increments.

## Agile backlog management and work assignment

[Jira](https://amanacu.atlassian.net/jira/software/projects/AT/boards/1) is used alongside [GitHub](https://github.com/TQS-Project-Org/Main-Repo) to track issues, manage projects, automate workflows and assign tasks to team members.

Our software development team follows a branch-driven development approach to ensure efficient and organized development processes. Each feature or issue is assigned a dedicated branch, allowing for isolated development and testing. After completing the development, code undergoes a thorough review process with collaboration and feedback within the branch. The changes are then integrated into the main branch, resolving any conflicts that may arise. Following successful testing, the updated code is deployed and released based on the merged branches, ensuring accurate tracking and version control. This branch-driven development methodology promotes productivity, code quality, and effective maintenance and support.

# Code quality management

## Guidelines for contributors (coding style)

For Java development, the [Google Java Style](https://google.github.io/styleguide/javaguide.html) guidelines will be followed to maintain a cohesive coding style.

Similarly, [Google JavaScript Style](https://google.github.io/styleguide/jsguide.html) will be used during the development of the React applications.

## Code quality metrics

SonarQube will be used for continuous inspection of code quality through static analysis.

The following quality gates are defined:

* Test coverage greater than 80%

[TODO]

# Continuous delivery pipeline (CI/CD)

## Development workflow

Code review practices are integrated into the project to guarantee code quality and foster collaboration. Through peer reviews and the utilization of resources such as checklists and style guides, issues are identified, and coding standards are upheld. This approach facilitates the creation of reliable and maintainable software.

A user story is completed when the following DoD (Definition of Done) criteria are met:

* Story approved by the Product Owner.
* Unit tests passed.
* Functional tests passed.
* Code reviewed.
* Non-functional requirements met.
* Documentation written.

## CI/CD pipeline and tools

We will utilize GitHub Actions for our CI/CD pipeline. We'll define stages like build, test, and deploy, and create YAML workflow files. Leveraging pre-built actions and customizing as needed, we'll automate tasks such as building, testing, and deploying our application. Regular monitoring and improvements will ensure efficient and reliable software delivery.

# Software testing

## Overall strategy for testing

We have decided to adopt Test Driven Development (TDD) as our primary development methodology. By following the TDD approach, we will create a set of tests that define the desired behavior of units of functionality before implementing the corresponding code. TDD will serve as a foundational practice in our development workflow, empowering us to build reliable and robust software.

In conjuction to TDD, we’ve also commited to Behavior Driven Development (BDD), with Cucumber as our preferred framework to enhance our development process.. By focusing on defining clear behavior specifications and creating executable scenarios, BDD enables us to prioritize user-centric perspectives and ensure that our software meets the desired behavior and outcomes. We’ll use Cucumber to map user stories to feature definitions.

## Functional testing/acceptance

[Project policy for writing functional tests (closed box, user perspective) and associated resources.]

## Unit tests

As outlined in section 4.1, Unit testing will follow TDD procedures, meaning that only the minimum amount of functionality for the module will be written before the corresponding test, in a way that the test will successfuly run but fail nonetheless.

As these tests are wirrten with the develoeprs perspective in mind, they configure as open box.

## System and integration testing

[Project policy for writing integration tests (open or closed box, developer perspective) and associated resources.]

API testing