Software Quality Assurance Plan (v. 1.1)

Company 1 - TDDC88

Contents

Version	Author	Updates	Reviewed by	Date
0.1	Erik Sköld	Initial version	-	2021-09-14
1.0	Erik Sköld	First plans for documents, development, testing and quality	Daniel Ma	2021-09-23
1.1	Erik Sköld	Updated guidelines for development regarding workflow, software reviews and version handling	Jacob Karlén	2021-10-13

1 Purpose

The purpose of this Software Quality Assurance Plan is to establish the processes and responsibilities required to implement effective quality assurance functions for the project.

The Software Quality Assurance Plan will provide the framework necessary to ensure a consistent approach to software quality assurance throughout the project life cycle. It defines the approach that will be used by members of the company to monitor and assess software development processes to provide objective insight into the quality of the software and the product. The systematic monitoring of the product and processes will be evaluated to ensure they meet requirements.

1.1 Scope

This plan covers Software Quality Assurance activities throughout the first iterations of the project and will be updated regularly.

1.2 Project description

The project's goal is to develop a tool that visualizes and compiles information from patients at Region Östergötland hospital's emergency departments. The information comes from the various computer systems that Region Östergötland has available today. The tool should compile information in an easy and clear way. The compiled information must be presented but does not have to include input solutions.

The product will be developed during four iterations. The project will continue until December 16, 2021, when version 1.0 will be presented.

2 Documents

This section lists the regulatory documents used in the project and general guidelines for documentation.

2.1 Regulatory documents

The guidelines described in this section apply to the seven regulatory documents listed below. *Links to the documents will be available in a future version of this document.*

- Architecture Notebook
- Customer Requirements Specification
- Education Plan
- Project Plan
- Software Quality Assurance Plan *(this document)*
- Standard Operating Procedures
- Test Plan

2.2 Guidelines

- Regulatory document shall be written in LaTeX, preferably with the help of Overleaf. The documents are made available to everyone in the company through link sharing, editing links to all documents are among the company's files in Teams.
- Version control of the regulatory documents is made via GitLab. Each document has its own directory in the documents folder of the develop branch. Version control shall be carried out regularly, at least when each new version of the document is published.
- To ensure traceability: When refering to requirements, issues or other artifacts in documentation, ID of the artifact in question must be included. When documenting actions, such as decisions or changes, pointers to relevant documentation must be included.
- For major changes to be made to requirements, a change request *(process TBD)* must be accepted by the product owner and reviewed by a company member with knowledge of the subject. Major changes must be documented in accordance with *(process TBD)* where the requirements and issues involved are clearly referred to.
- In order to publish a new versions of a regulatory documents, a company member with knowledge of the subject must have reviewed the updated document.
- When changes are made to published documents, version number, author, significant changes between versions, reviewer and date must be noted in the version table.

3 Development

This section is intended to address guidelines for the development and quality assurance of code that will result in the end product. The project is set up with TypeScript. The client (Angular), server (Node.js) and database (MongoDB) run in three seperate Docker containers and are orchestrated with Docker-compose.

3.1 Code conventions

As the project is set up with TypeScript (TS), Google's TypeScript Style Guide must be followed for all TS code.

Names for functions and variables must be written with camelCase and be self-explanatory. Names generated by Angular should not be changed. *Conventions for HTML and CSS is to be determined.*

3.2 Workflow

The project uses a form of feature branch workflow. Feature branches are merged with a development branch, never directly to the main branch. Once the code has passed testing on the develop branch, it can be merged into the main branch by an authorized manager.

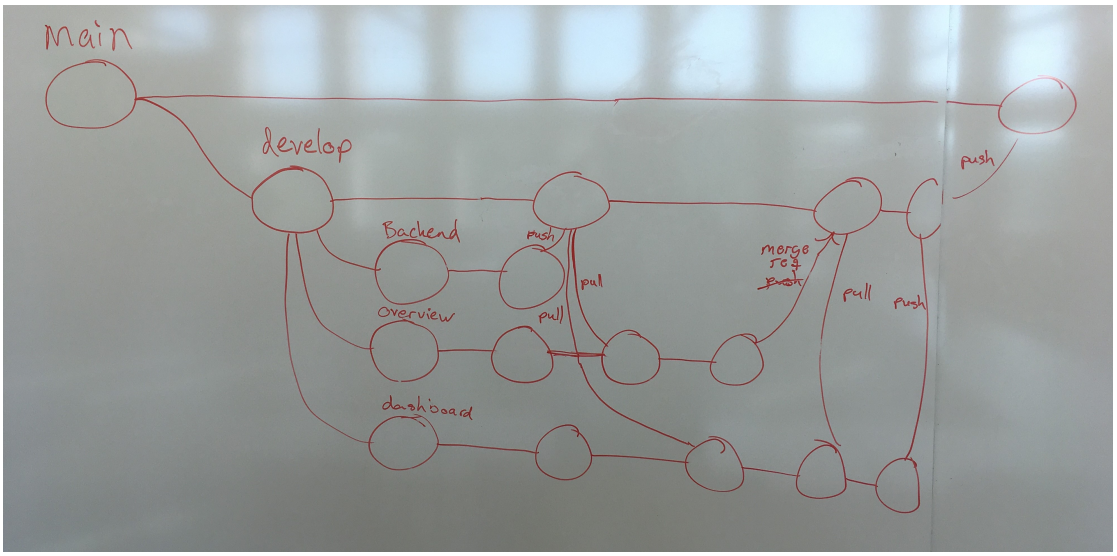

Figure 1: Sketch of current workflow on GitLab.

There are five long-lived branches:

- *main* - Is updated each iteration, code on this branch must be able to be shown to the customer. Members are only allowed to pull code from develop to main.
- *develop* - The intermediate branch where code from backend, overview and dashboard should be merged to. Code on this branch must be ready to be tested (integration, regression and acceptance testing). Here it is fine to have some bugs.
- *backend* - CFT1
- *overview* - CFT2
- *dashboard* - CFT3

From the three cross-functional team branches, smaller feature branches shall be created. These smaller branches are not supposed to live longer than maximum two days, this is to

enable continuous integration. When these smaller branches are merged to their respective CFT branch, peer-reviews must be made (see subsection 3.3 below). All merge requests must be approved according to the set guidelines before it is merged.

3.3 Software reviews

In addition to regular discussion of code between developers, software reviews take place in connection with each merge from feature branches to the development branch. The software review must follow a set protocol in GitLab, the guidelines can be found in the README-file on GitLab. In merge requests, the related requirement must be declared as well as what new features have added, the expected behavior of these and what to test.

The reviews are done by a project member who has not developed the code themselves within two working days of when the merge request was posted. The review must not exceed one hour. For merges to the main branch, the review must be done by the project architect.

3.4 Bug-tracking system

GitLab is used for bug-tracking. Bugs will be listed, classified and get unique ID as issues in the Issues tab of the company GitLab. *Further guidelines for bug-tracking will be found on the company GitLab.*

3.5 Version handling

GitLab is used for version handling. Work needs to be commited at least daily when working on features but the goal is to commit after each working addition to the code. Commits must be tagged with what kind of commit it is (see the section Semantic Commit Messages in the README-file on GitLab).

3.6 Documentation

The Technical Writer and the involved developers document the code produced at module level. The documentation includes descriptions of which inputs are handled, which functions are used, which inputs are given and how they work together with the rest of the system.

4 Testing

The Quality Coordinator will assure that the test management processes and products are being implemented per the Test Plan. This includes all types of testing of software system components as described in the test plan, specifically during integration testing (verification) and acceptance testing (validation).

The Quality Coordinator together with the Test Leader will monitor testing efforts to assure that test schedules are adhered to and maintained to reflect an accurate progression of the testing activities. It will be assured that tests are conducted using approved test procedures and appropriate test tools, and that test anomalies are identified, documented, addressed, and tracked to closure.

Members of the Test Team will review post-test execution related artifacts including test results and test reports.

4.1 Purpose

Testing will be conducted in order to verify that the product is working as intended. Solutions will be evaluated to reduce the risk of problems and improve the performance of the product. Testing is an important part of the work to ensure that the requirements are complied with and thus assuring quality.

4.2 Testplan

Link to Test documents will be added in an upcoming version.

5 Quality

This section highlights the expected resources to be spent on Quality Assurance (excluding testing) and the standard metrics to be used. *Will be further developed in a future version.*

5.1 Resources

Quality Personnel	Hours
Quality Coordinator	80

5.2 Standard metrics

- Software Quality effort and resources expended (Planned versus Actual)
- Number of Software Quality Assessments (Planned versus Actual)
- Number of Software Quality Assessment Findings or Noncompliances (Open versus Closed)
- Number of Risks identified as a result of Software Quality Assessments