Technical Implementation of a Software Product

Course:	4A00EZ64-3003/3004 Monialainen projektityö
Product:	Process-tracking tool prototype
When implemented?	21.10.2025 - 12.12.2025
Members of the project team:	Nitai Spira, Henri Nieminen, Noora Nevalainen, Tuuli Marttila, Santeri Sillanaukee
Client/customer:	L1VE GmbH

Version History of this Document		
Date	**Modified by**	**Description of Changes**
28.11.2025	Nitai, Henri, Noora, Tuuli, Santeri	The first draft
		(add rows as necessary)

1 Product

1.1 Product Description

This product is a deployable web application to track processes and its related tasks.

A concise description about the product and its purpose.

1.2 Other Documentation

README.md file is in the code repository.

References/addresses to other relevant documents.

1.3 Code etc. Availability

The code is handed over as part of the deliverables for the project. Storing and hosting the code is the responsibility of the client

1.4 Relevant Addresses to Access the Product

Currently there is no permanent address to access the product. It is up to the client to deploy the application and to get any needed domains to properly host it.

For web-based products, addresses (links) to possible applications running in different environments (production/development/test/other).

2 Implementational Decisions Made

2.1 Constraints, Demands, and Justifications for the Decisions

The application has been developed to be deployed as a web application for easy accessibility and flexibility. This ensures flexible hosting and seamless expansion in e.g cloud services.

Note that the app is currently designed to be used on a computer, and will not work properly on devices without a sufficiently large screen.

In which kind of an operating environment have the decisions been made? Which matters one has been forced to take into account? (E.g., legal issues, standards, internal processes/policies of the customer, other customer requirements, affordability / allowed expenses, availability, expected strain due to the anticipated usage...) Why has one ended up with the decisions that have been made? Document also possible solutions / implementational approaches that have been tested but found to be suboptimal/infeasible/bad (and why)!

2.2 Languages, Libraries, and Tools Used

The backend consists of Java and Spring Boot. Database architecture leans on MariaDB.

Javascript, along with React and Vite, are used to build the frontend.

Backend libraries used:

- JUnit
- H2
- Lombok
- MariaDB Java client

Frontend libraries used:

- React
- Vite
- Axios
- React Router
- React Toastify
- Material UI
- React Flow

3 User Roles

The product has no role functions due to time constraints.

4 Data Protection and Data Security

The product does not contain any sensitive data from the start, but it can contain operation sensitive data in the future when the customer implements it. Backend code and Spring Boot is made sure to work well with not letting any lousy inputs get through, but it is up to the customer to make sure their version of the database is properly secured.

"How have been and are data protection and data security taken care of? Does the system contain sensitive data? Are there special risks or updating needs that should be known to

maintainers and (future) developers? How have taking and storing backup copies been arranged? Relevant references to possible other documentation.”

5 Architectural Overview of the System / Operating Environment

Primary use of the program is within a Docker container that can be deployed on a local server or cloud services. We utilized Höchshule Munchen’s AWS platform to test our program.

The development version uses an in-memory database to store data, but the production version will need a MariaDB database. The database needs to be set up and configured by the customer first and the properties file in the backend code must be properly linked to access the database.

Description of the parts of the system and their relations. What is run using which servers and how? Which component / piece of software / application communicates with which, in which networks, and how? Where is information retrieved from and where is it stored to? A figure (or several figures) to make it easier to understand the whole and the relevant relations. Reasons and justifications.

6 Code Structure

Frontend:

- **Base path:** <project-root>/frontend/JSFrontend/
- Frontend code is contained in <base>/src
- React components are stored in <base>/src/components
- React contexts are stored in <base>/src/Context
- CSS stylesheets are stored in <base>/src/style

Backend:

- **Base path:** <project-root>\backend\proseed\src\main\java\com\proseed
- Controllers for API endpoints are stored in <base>\controllers
- Database Entities are stored in <base>\entities
- Database interaction files are stored in <base>\services\impl
- Data Transfer Objects are stored in <base>\DTOs
- DTO Mappers are stored in …\DTOs\Mappers

What are the essential parts of the product as far as the code is considered? In which files have they been implemented? Have some known architectural code patterns or library functionalities been used? Is there something special and noteworthy concerning the code or related data? E.g., class diagrams can be used to demonstrate the code structure.

7 Data Storage Solutions

This application uses Spring Boot with MariaDB schema.

A sufficiently detailed description on, e.g., database management systems – with version information – and databases or such. (E.g., DB table structure / creation commands).

8 Maintenance

8.1 Responsibilities and Maintenance Processes

The responsibility would befall on the IT department of the client. At its current state, the application cannot hold any more personal data than just legal name, department and skills. These can be deleted swiftly in the frontend.

Who is responsible for what when the system is run/maintained? On which organizational protocols or processes is this division based? (E.g., if an end user requires his/her user information to be removed from personal data registries, is there in use some clear and documented process about how to handle this? How is this reflected on the maintenance work?)

8.2 Installation

Installation of the system is done by building the Docker image from the source and then running and exposing the ports in the system to allow traffic into it. The default port for the app is 8080, but it can be changed later.

How is the system (the parts of it) installed "from the scratch"? Environment requirements? The instructions must be detailed enough so that one can get the system up and running based on them.

8.3 Stopping and Starting the Operations, Making Backup Copies and Restoring the System, and Emergency Procedures

Backups of the database must be done manually or automation configured if hosted locally. If hosted in cloud services, consult the cloud services for automated backups. AWS for instance can help with and hold backups.

See the README.md file for further instructions in the codebase.

Instructions to execute the typical/critical operations within the scope of maintenance (the actual commands required with necessary explanations).

8.4 Modifying the System

The most crucial part to keep in mind in modifying the system, is to be mindful of the database schema. If you want to expand more features, you may need to expand the database with required fields.

The issues and matters that should be taken into account when changing the code or data. The procedure to follow in order to deploy the modified version of the product. Sufficiently detailed instructions.

8.5 Other Upkeep and Management Needs

Upkeep must be done on a semi-frequent basis. The project uses NPM package manager on the frontend side, and uses Gradle in the backend.

To avoid any security mishaps or vulnerabilities, be mindful of updating the packages every now and then, and possibly update the code in any major package version changes.

E.g., known time-based needs for making changes or applying updates, adding and removing users, etc. Sufficiently detailed instructions.Logging, Reports, and Monitoring

8.6 Logging, reports, and monitoring

The application currently does not save logs to a separate file.

Are log data gathered? What information is logged and where is it stored? Is the system capable of producing reports on its operation? (How to obtain/use them?) Are there automated monitoring or alert features?

9 Instructions for Basic Usage

See README.md file in the repository.

User's guide/manual, if not published as a separate document (or as a part of the system UI). Pictures (screen captures) may be useful – clarity is important. If the manual can be found separately, the reference should be included here along with possible additional information and observations.

10 Missing Features, Known Problems, and Future Development

Missing features:

- User interface design and scalability for mobile devices
- Authentication/User roles
- Assigning employees to tasks
- Assigning departments to tasks
- Assigning skills to tasks
- Task status states
- Color coding based on departments

Known problems:

- Tasks with children cannot be deleted
- New tasks cannot be inserted between existing task and process

List the planned features still to be implemented. Are there still needs for additional testing? Are there known bugs or "interesting features"? Are there plans for future development? Are

there matters possible future developers should be aware of or pay special attention to? Suggestions for improvement (not mentioned on the list of missing features)?