# Technical Implementation of a Software Product

| | |
|---|---|
| **Course:** | 4A00EZ64-3003/3004 Monialainen projektityö |
| **Product:** | ProSeed |
| **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 |
| 2.12.2025 | Nitai, Henri, Noora, Tuuli, Santeri | Second draft |
| 10.12.2025 | Nitai, Henri, Noora, Tuuli, Santeri | Final draft |

# 1 Product

## 1.1 Product Description

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

## 1.2 Other Documentation

README and user manual are stored in the code repository.
User interface and user experience design for the web application was done in Figma: Figma prototype

## 1.3 Code etc. Availability

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

## 1.4 Relevant Addresses to Access the Product

Currently there is no permanent address to access the product. The application is temporarily available on AWS at http://13.48.127.123:8080/. It is the responsibility of the client to deploy the application and to get any needed domains to properly host it.

# 2 Implementational Decisions Made

## 2.1 Constraints, Demands, and Justifications for the Decisions

Most, if not all decisions were made based on the wishes and needs of the client, and available time.

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. There was no budget allocated to the project, and all libraries and software used in the project are available for free..

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.

## 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 languages, libraries and tools used:

- Java v21.0.9
- Gradle v8.14.3
- Spring Boot v3.5.7
- JUnit v*
- H2 v*
- Lombok v*
- MariaDB Java client v*

Frontend languages, libraries and tools used:

- Javascript
- NPM v11.6.3
- React v19.1.1
- Vite v7.1.7
- Axios v1.13.1
- React Router v7.9.4
- React Toastify v11.0.5
- Material UI v7.3.4
- React Flow v12.9.2

Libraries marked with * use the latest version fetched by Gradle.

# 3 User Roles

Endpoints for performing CRUD operations on roles are implemented and roles can be added to individual employees in the database as part of a PATCH request. There is currently no other functionality associated with roles.

# 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.

# 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 must be properly linked to access the database.

The frontend is a React web application, while the backend is a Spring Boot server that contains the database that will store all the information that is used in the application. In development mode, the front and backend are separated, but when the application has been built and is running in production mode, the backend will serve the frontend as static content.

Operations in the application are performed through sending HTTP requests from the frontend to the backend. The backend provides endpoints for different operations, such as fetching and CRUD operations.

# 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/

ProcessContext provides functions for handling processes, such as CRUD functions. TaskContext is similar to ProcessContext, but for tasks. DataContext stores processes and tasks in the state and provides functions for fetching processes and tasks and for handling operations that need functionality from both ProcessContext and TaskContext.

Backend:

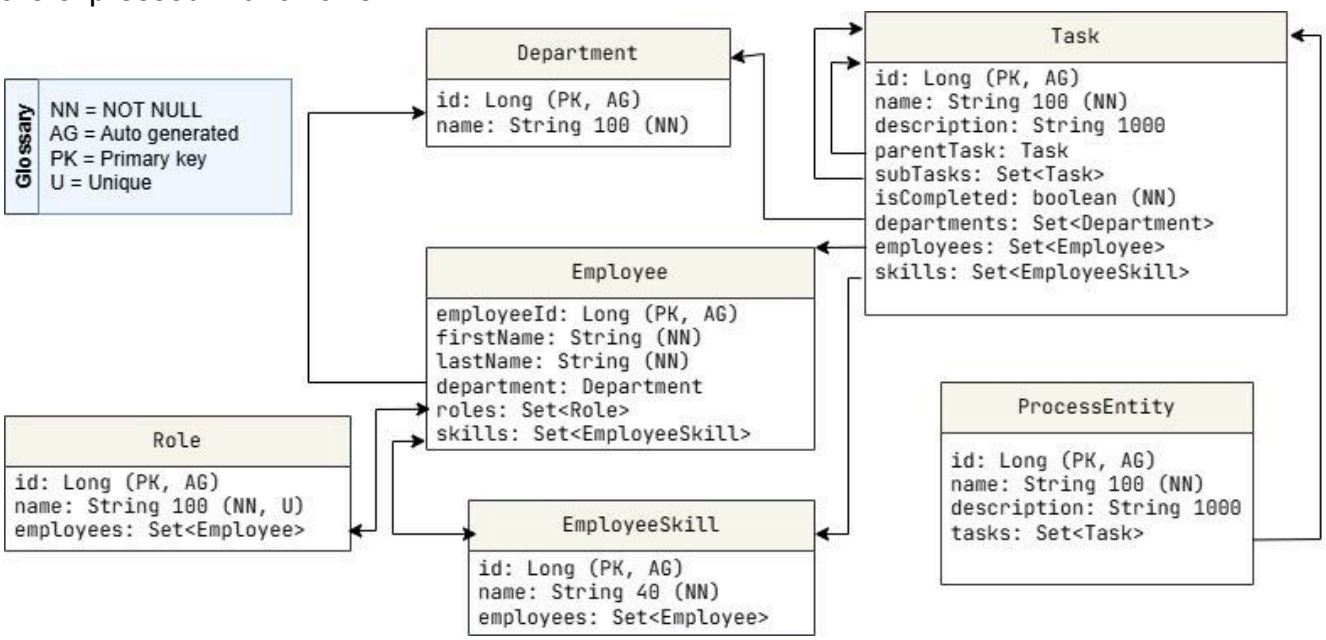
- **Base path:** <project-root>/backend/proseed/src/main/java/com/proseed
- Controllers for API endpoints are stored in <base>/controllers
- Database Entities are stored in <base>/entities
- Database interaction files are stored in <base>/services/impl
- Data Transfer Objects are stored in <base>/DTOs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

# 7 Data Storage Solutions

The backend uses the following dependencies to implement the database:

- Spring Boot v3.5.7
- MariaDB v*
- H2 v*

This application uses Spring Boot with MariaDB schema.
Spring Boot creates database structure using Hibernate based on classes in entities-folder. See the database Class diagram below, where entity structure is shown and entity relations are expressed with arrows.



Libraries marked with * use the latest version fetched by Gradle.

# 8 Maintenance

## 8.1 Responsibilities and Maintenance Processes

Maintenance of the software will be the responsibility of the IT department of the client, or the people hired to maintain and/or further develop the software. 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.

## 8.2 Installation

**Production**:

The only dependency for production use is Docker, which is required to be installed to build and run the application inside a Docker container. Follow the instructions for installing Docker found in the Docker documentation https://docs.docker.com/desktop/. Instructions for building and running the Docker container are in the README file in the code repository.

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.

**Development:**

Development of the software requires the following dependencies:
Frontend development:

- NodeJS

Instructions for installing NodeJS and NPM: https://nodejs.org/en/download

NOTE: Make sure to select NPM as the package manager for NodeJS, as the project was developed using NPM, and documentation was written with the assumption that NPM is used.

Backend development:

- Java 21
- Gradle 8.14.3

SDKMAN is recommended if installing Java and Gradle through the terminal.
Instructions for installing SDKMAN: https://sdkman.io/install/
Instructions for installing Java and Gradle through the terminal using SDKMAN:
Java:

```
sdk install java 21.0.9-tem
```

Gradle:

```
sdk install gradle 8.14.3
```

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

See the README.md file for further instructions on how to start the application locally in a docker container

Backups of the database must be done manually or automation configured if hosted locally. If hosted in cloud services, consult the cloud service provider's documentation for implementing automated backups.

## 8.4 Modifying the System

When updating the database entities, make sure to update existing repository implementations to account for changes in the structure. Any corresponding Data Transfer Objects will also require updates to reflect the new state of the database, if the changes interact with the API.

## 8.5 Other Upkeep and Management Needs

Upkeep should 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 occasionally, and possibly update the code in any major package version changes.

### 8.6 Logging, reports, and monitoring

The application currently does not save logs to a separate file. The frontend will show an error toast when an error occurs during backend operations.

## 9 Instructions for Basic Usage

See MANUAL.md file in the code repository for the user manual.

## 10 Missing Features, Known Problems, and Future Development

Future development ideas:

- User interface design and scalability for mobile devices
- Authentication/User roles
- Assigning employees to tasks
- Task status states
- Color coding of the different departments
- Color coding task nodes by responsible department
- Search departments/skills by synonyms

Known issues:

- Tasks with children cannot be deleted
- New tasks cannot be inserted between existing task and process
- Attempting to delete the first or second skill tag results in an error. This only seems to affect the first two skills and only when they’re created on startup i.e. if no skills are initialized on startup, and you create a new skill, it can be deleted successfully. As such this should only happen if the application is started in development mode, as dev mode initializes some dummy data