Customer Requirements Specification

(Lastenheft)

(TINF19C, SWE I Praxisprojekt 2020/2021)

Project: *AML NoSQL Database Management*

Customer: *Rentschler & Holder*
Rotebühlplatz 41
70178 Stuttgart

Supplier: Team 5:
(Nils-Christopher Wiesenauer, Namid Marxen, Johannes Timter, Jonas Bihr, ~~Max Scheub~~)
Rotebühlplatz 41
70178 Stuttgart

Version	Date	Author	Comment
0.1	21.10.2020	Timter	Edit Goal
0.2	22.10.2020	Marxen	Edit Features
0.3	22.10.2020	Wiesenauer	Edit Product Enviroment
0.4	24.10.2020	Team	Working on everything at “Hackathon”
0.5	03.11.2020	Marxen	Changed Business Process
0.6	05.11.2020	Marxen	Changed Product Data
1.0	09.11.2020	Marxen	Finished Overhaul

CONTENTS

1. Goal

Automation **M**arkup **L**anguage (AML) is an XML based data format to store and share engineering data about parts of production plants. These objects can consist of multiple other objects and be part of a larger assembly of objects. For example, an AML document can describe a screw, a claw, a robot or a complete manufacturing cell in different levels of detail. [3]

It is the goal of this project to develop a web application that allows the user to perform the **CRUD** (Create, Read, Update, Delete) operations on JSON converted AML files. The functions of the web application must be accessed through a graphical user interface. More precisely, the user should have the options to upload and download AML files and search for existing AML files with an ID-based search field.

Implementation: The frontend should be implemented with Angular. The backend should be implemented with Express.js and Node.js. A local MongoDB instance should be used to store the JSON converted AML documents.
As AML is based on XML and MongoDB is only able to store JSON documents, it is necessary to model and implement a conversion function for our AML documents from and to the JSON format.

The finished product should allow engineers to share and access their AML documents with others. As we are implementing the product as a web application, the product will accessible anywhere, from any web-capable end device. The process of finding existing documents and uploading your own documents should be fast and easy, even for non-engineers. Therefore, our target-group are not only engineers, but anyone who wishes to inspect and understand AML documents.

2. Product Environment

Angular is a TypeScript based front-end framework which is published as open source software. This framework has been around for almost 10 years and since then countless adaptations have been made. The three pillars of Angular are TypeScript, RxJS and Zone.js. [1]

In simple terms, NodeJS is a JavaScript free and open source cross-platform for server-side programming that allows users to build network applications quickly. The runtime is intended for use outside of a browser context (i.e. running directly on a computer or server OS). As such, the environment omits browser-specific JavaScript APIs and adds support for more traditional OS APIs including HTTP and file system libraries. [4]
ExpressJS is the most popular Node web framework and is the underlying library for several other popular Node web frameworks. It provides mechanisms to write handlers for requests with different HTTP verbs at different URL paths (routes), to integrate with "view" rendering engines in order to generate responses by inserting data into templates, to set common web application settings like the port to use for connection, and the location of template that are used for rendering the response and to add additional request processing "middleware" at any point within the request handling pipeline. [5][6]

MongoDB is a document-oriented NoSQL database used for high volume data storage. Instead of using tables and rows as in the traditional relational databases, MongoDB makes use of collections and documents. Documents consist of key-value pairs which are the basic unit of data in MongoDB. [2]

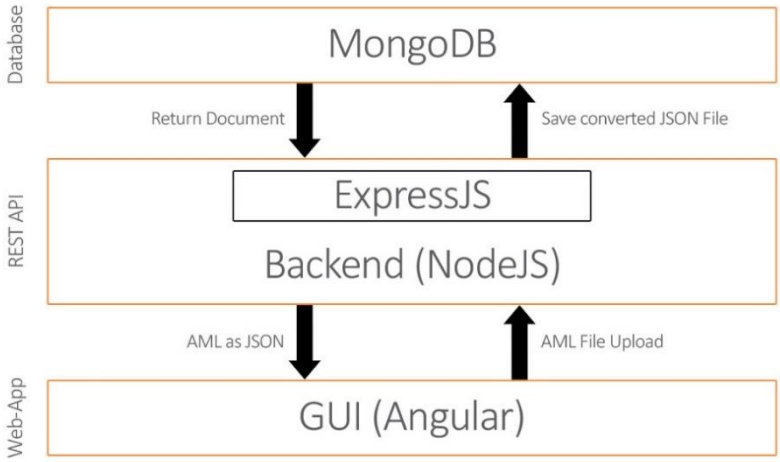

Figure 1 Product Environment

3. Product Usage

The main purpose of the software will be to upload AutomationML files to a database. The uploaded files can be accessed from the user through a web interface. In that interface the user is also able to edit, delete and download existing files. He should also have the ability to search for saved documents. This provides the user with an easy way to upload AutomationML files into a database and conveniently handle them through a web interface.

The following features should be accessible in the GUI of a web application:

1. The user can upload AML files
2. The user can edit existing AML documents
3. The user can delete existing AML documents
4. The user can download existing AML documents
5. The user can search for existing AML files through an ID-based search field

3.1. Business Processes

This section shows the individual business processes necessary to support the system to be developed.

3.1.1. <BP.001>: Share AML files using a web interface

Triggering Event:	User has an AutomationML file in XML format and wants to upload it to a database with a web interface.
Result:	The file is available to other users who use the web interface. They can then edit download and delete the file.
Involved Roles:	User and AMLDatabase

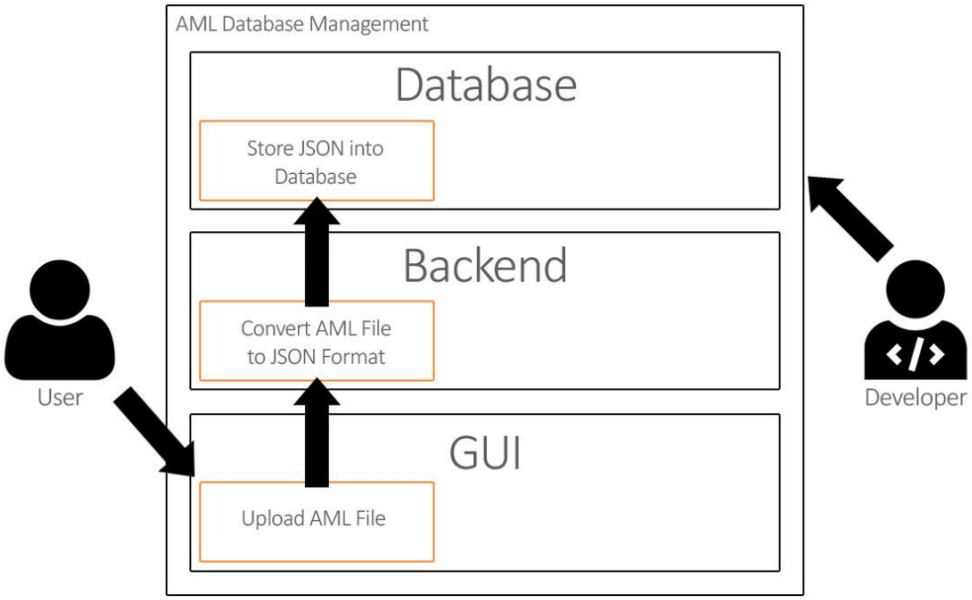

Figure 2 <BP.001> Share AML files

3.2. **Use Cases**

This project will be implemented with a graphical user interface. The user will have the opportunity to access the graphical user interface and upload AutomationML files in XML format, which will be saved into a database. He can then download, edit and delete the entries through the web-page. The user can also search for a saved file based on the ID.

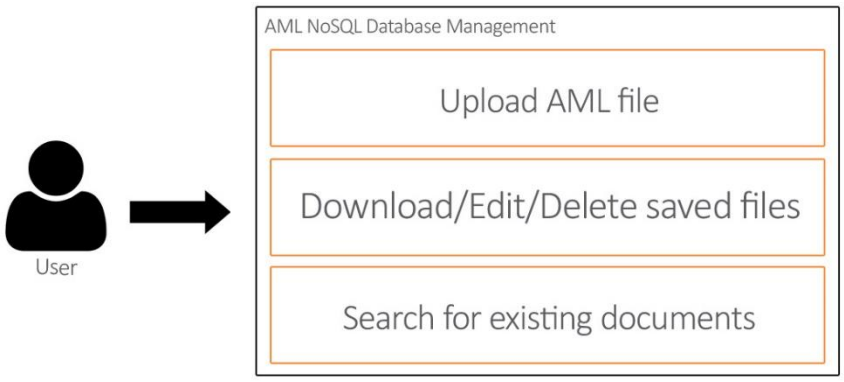

Figure 3 *Use Case Overview Diagram*

3.2.1. <UC.001> Upload AML files

Related Business Process:	<BP.001 >: Share AML files using a web interface
Use Cases Objective:	User wants to upload an AML file using a web-interface
System Boundary:	The web interface is the system boundary
Precondition:	- The web interface and backend server must be active - The file to be uploaded must be an AML file
Postcondition on success:	The page should not be closed while the user is uploading the file.
Involved roles:	User and AML Database web interface
Triggering Event:	Opening the web interface in the browser and selecting the upload function and then selecting a local AML file.

3.2.2. <UC.002> Search for existing files

Related Business Process:	<BP.001>: Share AML files using a web interface
Use Cases Objective:	User wants to search for an existing file in the web interface
System Boundary:	The web interface is the system boundary
Precondition:	- The web interface and backend server must be active
Postcondition on success:	The page should not be closed during the search process.
Involved roles:	User and AML Database web interface
Triggering Event:	Opening the web interface in the browser and selecting the search function and typing an id to be searched

3.2.3. <UC.003> Download files

Related Business Process:	<BP.001>: Share AML files using a web interface
Use Cases Objective:	User wants to download an existing file
System Boundary:	The web interface is the system boundary
Precondition:	- The web interface and backend server must be active
Postcondition on success:	The user does not cancel the download
Involved roles:	User and AML Database web interface
Triggering Event:	Opening the web interface in the browser and selecting a file and click on download.

3.2.4. <UC.004> Edit files

Related Business Process:	<BP.001>: Share AML files using a web interface
Use Cases Objective:	User wants to Edit a saved file
System Boundary:	The web interface is the system boundary
Precondition:	- The web interface and backend server must be active - The file is not being edited by another user
Postcondition on success:	The server does not close during the editing process
Involved roles:	User and AML Database web interface
Triggering Event:	Opening the web interface in the browser and selecting a file and clicking on edit.

3.2.5. <UC.005> Delete files

Related Business Process:	<BP.001>: Share AML files using a web interface
Use Cases Objective:	User wants to delete a saved file
System Boundary:	The web interface is the system boundary
Precondition:	- The web interface and backend server must be active
Postcondition on success:	The server does not close during the deletion process
Involved roles:	User and AML Database web interface
Triggering Event:	Opening the web interface in the browser and selecting a file and clicking on delete

3.3. Features

3.3.1. /F01/Graphical User Interface

The application shall support a graphical user interface which is easy to use.

3.3.2. /F02/REST API

The application shall implement a REST API for other developers to use.

3.3.3. /F03/Database

The application shall be able to save uploaded files to a database.

3.3.4. /F04/Upload files

The application shall be able to upload files selected locally from the user.

3.3.5. /F05/Conversion from XML to JSON

The system should be able to convert an uploaded AML file to JSON and save it in the database.

3.3.6. /F06/Download files

The application shall provide the user with the ability to download saved files from the database.

3.3.7. /F07/Edit files

The application should provide the user with the ability to edit the documents in the database with the graphical user interface.

3.3.8. /F08/Delete files

The application should provide the user with the ability to delete the documents in the database with the graphical user interface.

3.3.9. /F09/Search for files

The application should be able to search for existing files based on the ID, saved in the database.

3.3.10./F10/List all files

The graphical user interface should list all files from the database with sorting and filtering functionality.

4. Product Data

This section describes the various data the application uses.

4.1. /LD10/AML files

The data used in this application is based on the AML files the user uploads. These files are in the AML format which is based on XML. The files that the user wants to upload have to be in the AML format and the file has to have the extension .aml.

4.2. /LD20/JSON objects

The uploaded AML file has to be saved in a MongoDB database which is based on JSON. This is why the AML files have to be converted into JSON by writing the content of the file into one string which is saved as one JSON object. In the database, the entries are all JSON objects with one object, the string with the content of the AML files.

5. Other Product Characteristics

This section describes the already known non-functional requirements for the product.

5.1. /NF10/Usability

The system shall support a graphical user interface, which is intuitive and easy to use.

5.2. /NF20/Expendability

The system shall support a communication between frontend and the database, which should be developed in a modular way, to be easily expandable in the future.

5.3. /NF30/Data Integrity

The system should be able to save valid XML data in the database, while data integrity is provided.

5.4. /NF40/System Environment

The system should run under any operating system with a web browser installed.

6. References

[1] http://angular.io
[2] http://www.mongodb.com
[3] http://www.automationml.org/o.red.c/home.html
[4] http://nodejs.org/en
[5] http://expressjs.com
[6] http://developer.mozilla.org/en-US/docs/Learn/server-side/express_nodejs/Introduction