System Requirements Specification

(Pflichtenheft)

(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	16.10.2020	Namid Marxen	created
0.2	24.11.2020	Namid Marxen	added Use cases
0.3	02.11.2020	Namid Marxen	added Product Requirements
0.4	04.11.2020	Namid Marxen	added Product Data
0.5	05.11.2020	Namid Marxen	changed Product Data
0.6	09.11.2020	Namid Marxen	Complete overhaul
1.0	10.11.2020	Namid Marxen	Final version

CONTENTS

1. Introduction

This software is targeted at users, who want to manage all their AML files in one place, to make them easily accessible and saved securely on a database. The finished product should allow engineers to share and access their AML documents with others. The files are selected from the user's computer and then uploaded with a web-interface. As we are implementing the product as a web application, the product will be 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 any person who wishes to inspect AML documents. This is why a user-friendly web-interface is important for this project.

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 AML files. These functions must be accessible in the graphical user interface in form of the web application. To achieve this, the user will see a list of all files in the database where he can click on one of the files, to either delete, edit or download the entry. He also has the ability to upload files to the database and search for existing entries with an ID-based search field.

1.1. Product Environment

The frontend should be implemented with Angular. The Backend should be implemented with ExpressJS and NodeJS. A MongoDB database should be used to store the AML documents in JSON format.
As AML is based on XML and MongoDB is only able to store JSON files, it is necessary to model and implement a conversion function for our AML documents from and to the JSON format.

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]

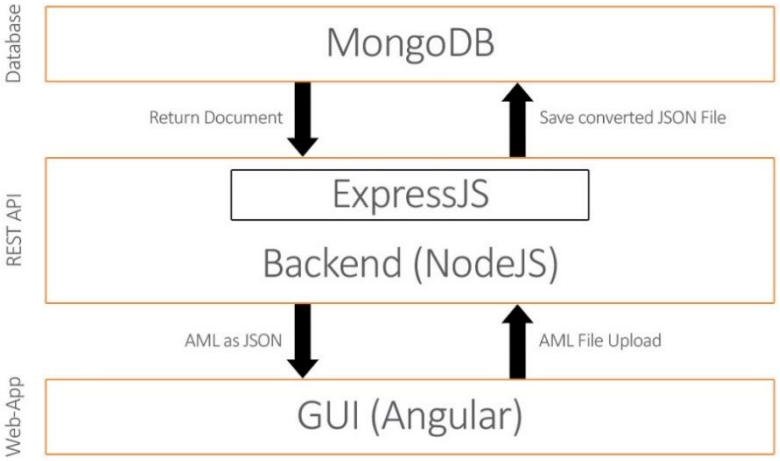

Figure 1 Product Environment

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]

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

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]

1.2. Use Cases

This project will be implemented with a graphical user interface. The user will have the ability 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.

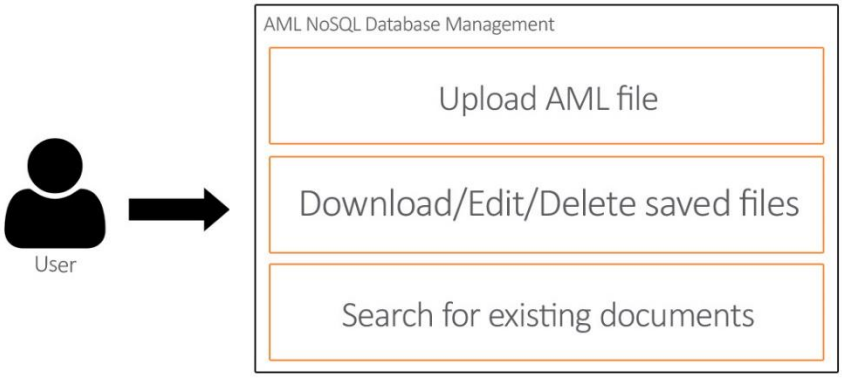

Figure 2 - Use Case Overview Diagram

1.2.1. <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, clicking on the upload button and then selecting a local AML file.*

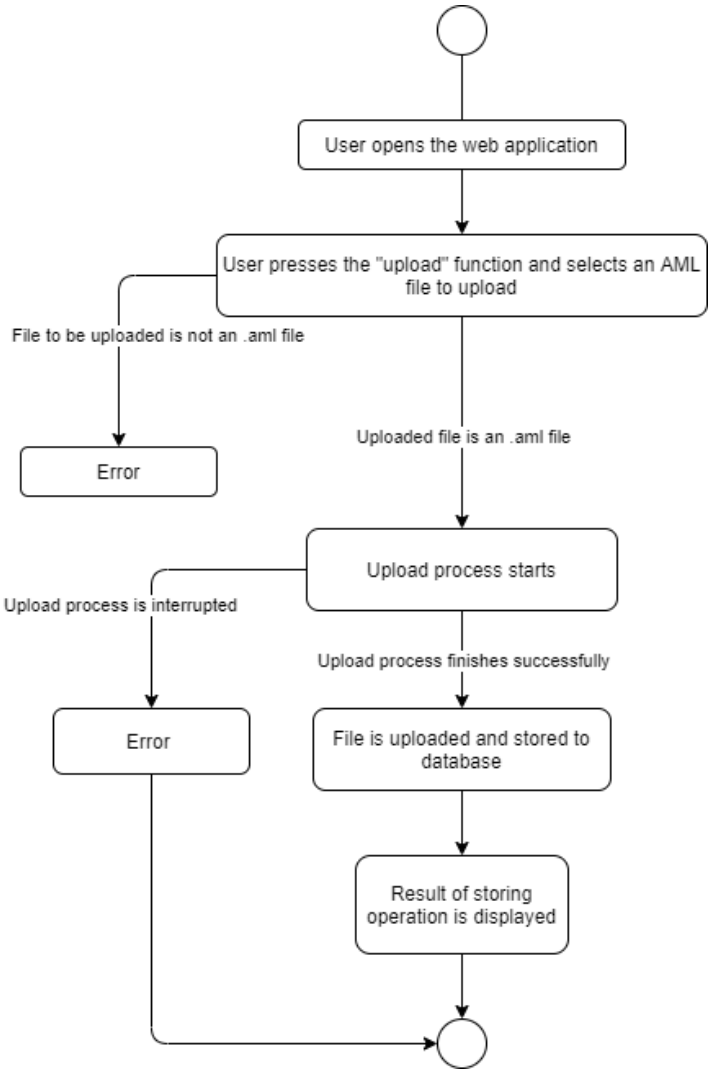

Figure 3 <UC.001> Upload AML files

1.2.2. <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, typing in the search input field and pressing enter, to confirm to input*

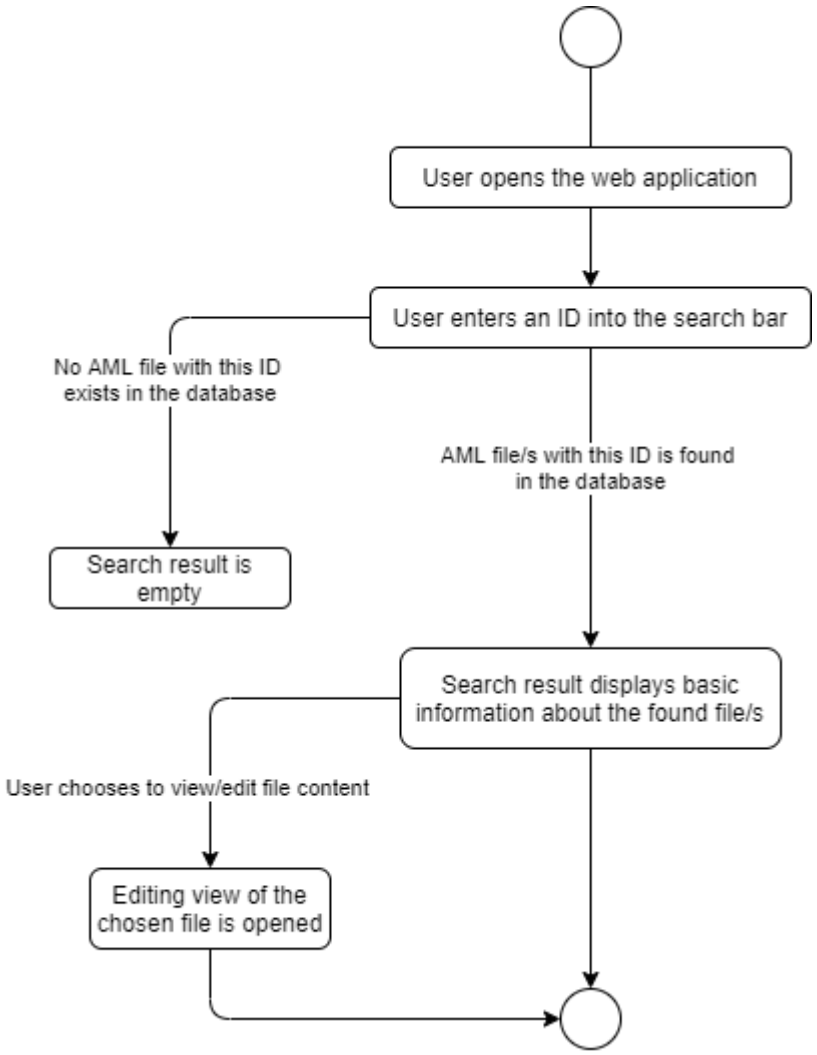

Figure 4 <UC.002> Search for existing files

1.2.3. <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 clicking on the download button on one of the listed files*

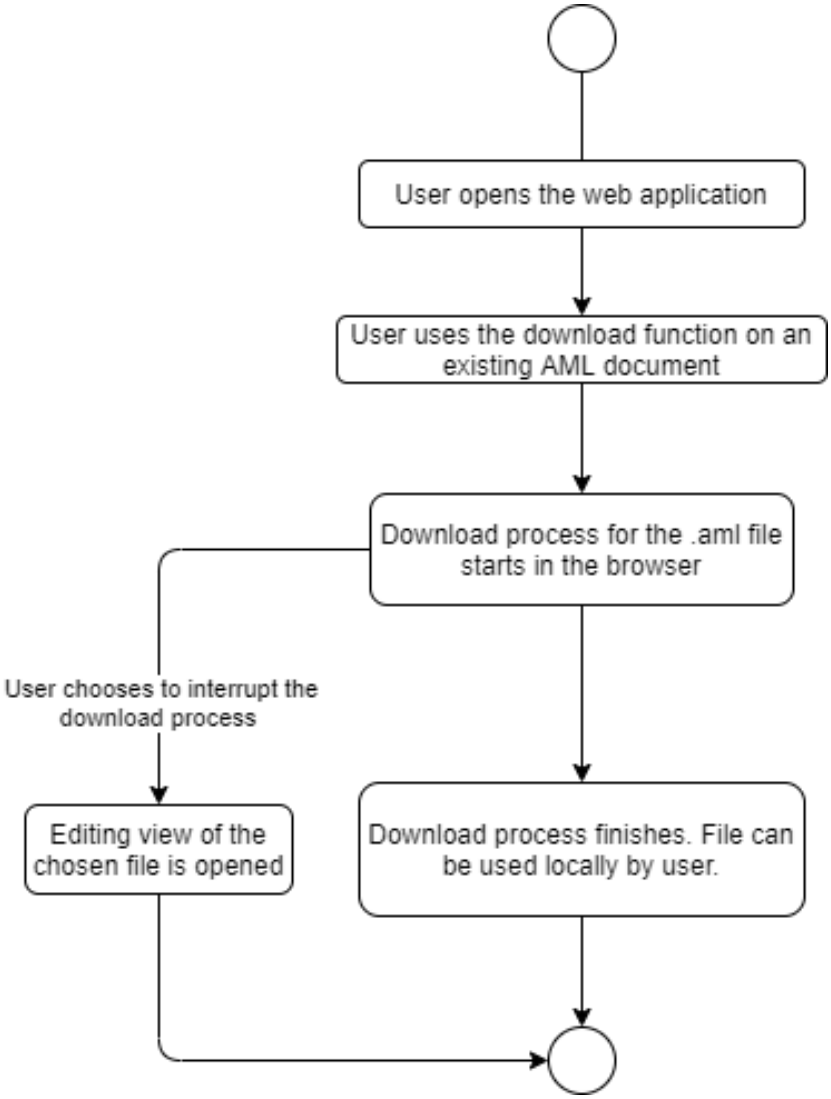

Figure 5 <UC.003> Download files

1.2.4. <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*
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 clicking on the edit button on one of the listed files*

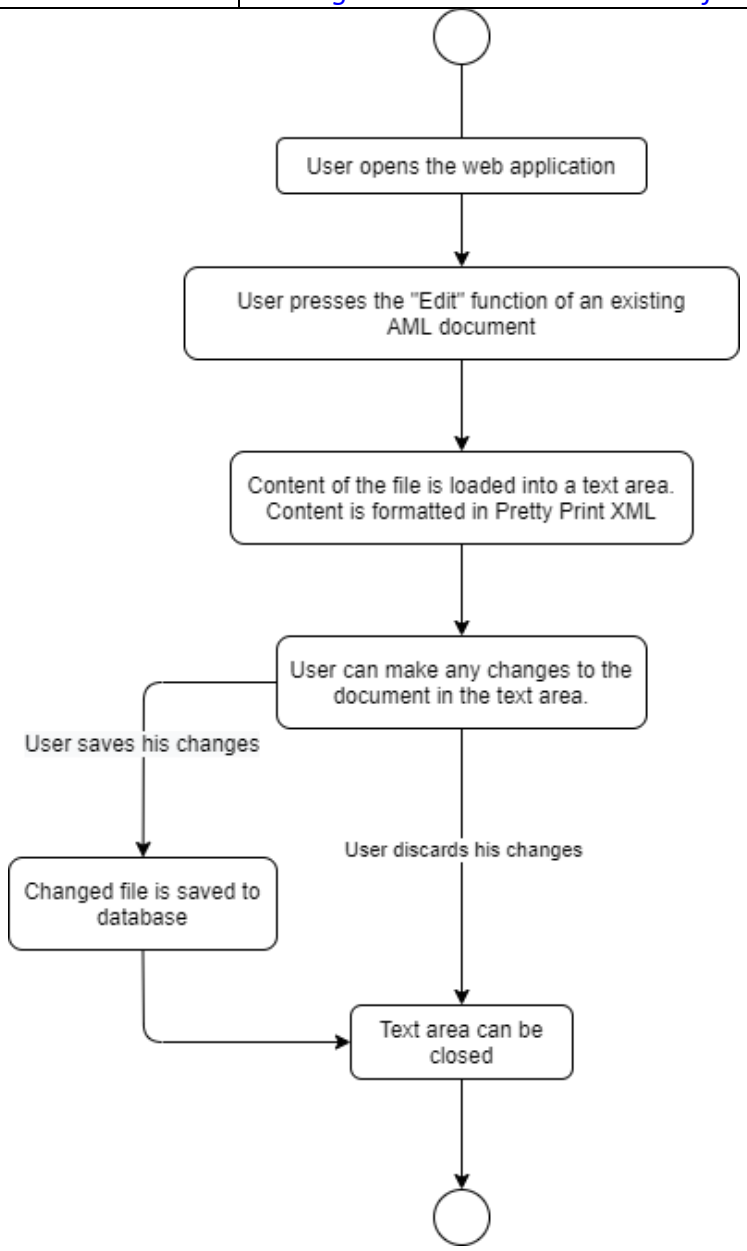

Figure 6 <UC.004> Edit files

1.2.5. <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 clicking on the delete button on one of the listed files*

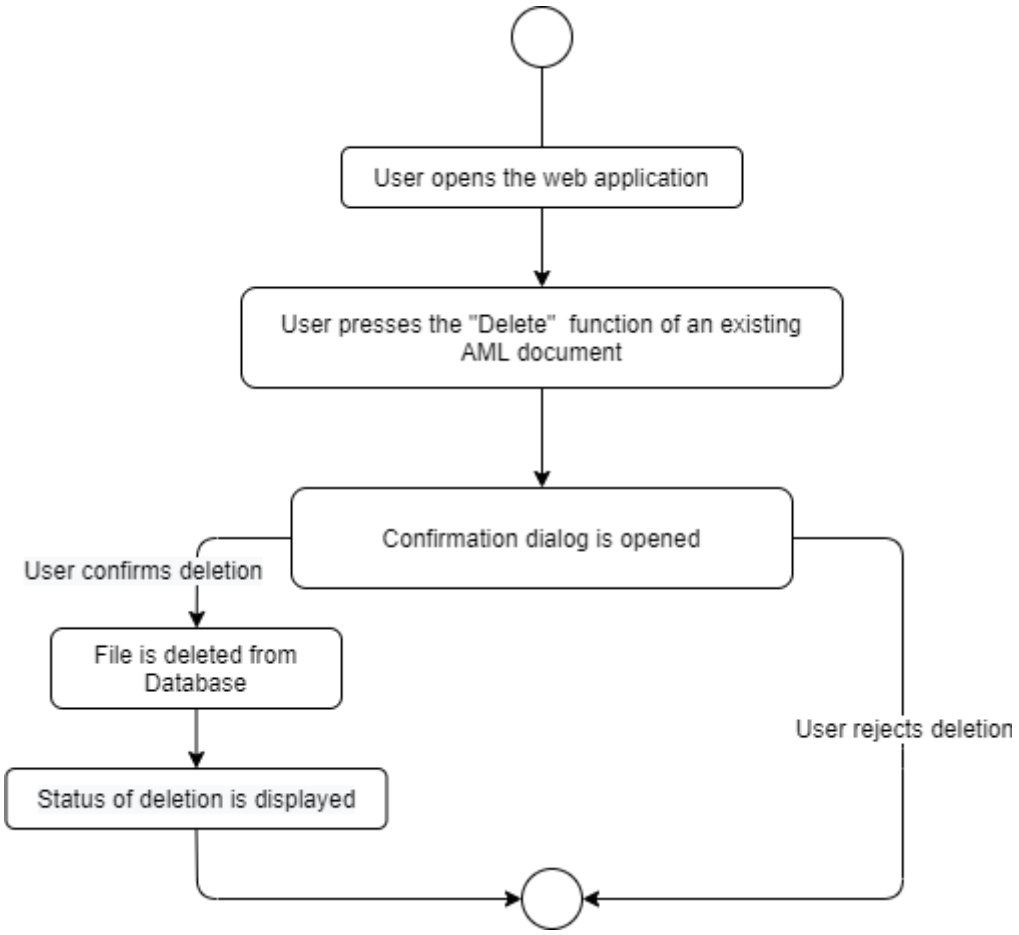

Figure 7 <UC.005> Delete files

2. Product Requirements

The following functionalities shall be supported by the system.

2.1. /LF10/Graphical User Interface

The user is able to interact with the system via a graphical user interface in form of a web-interface based on Angular. It shall provide the user with information about the saved files and the ability to edit, delete, download or upload files with the GUI.

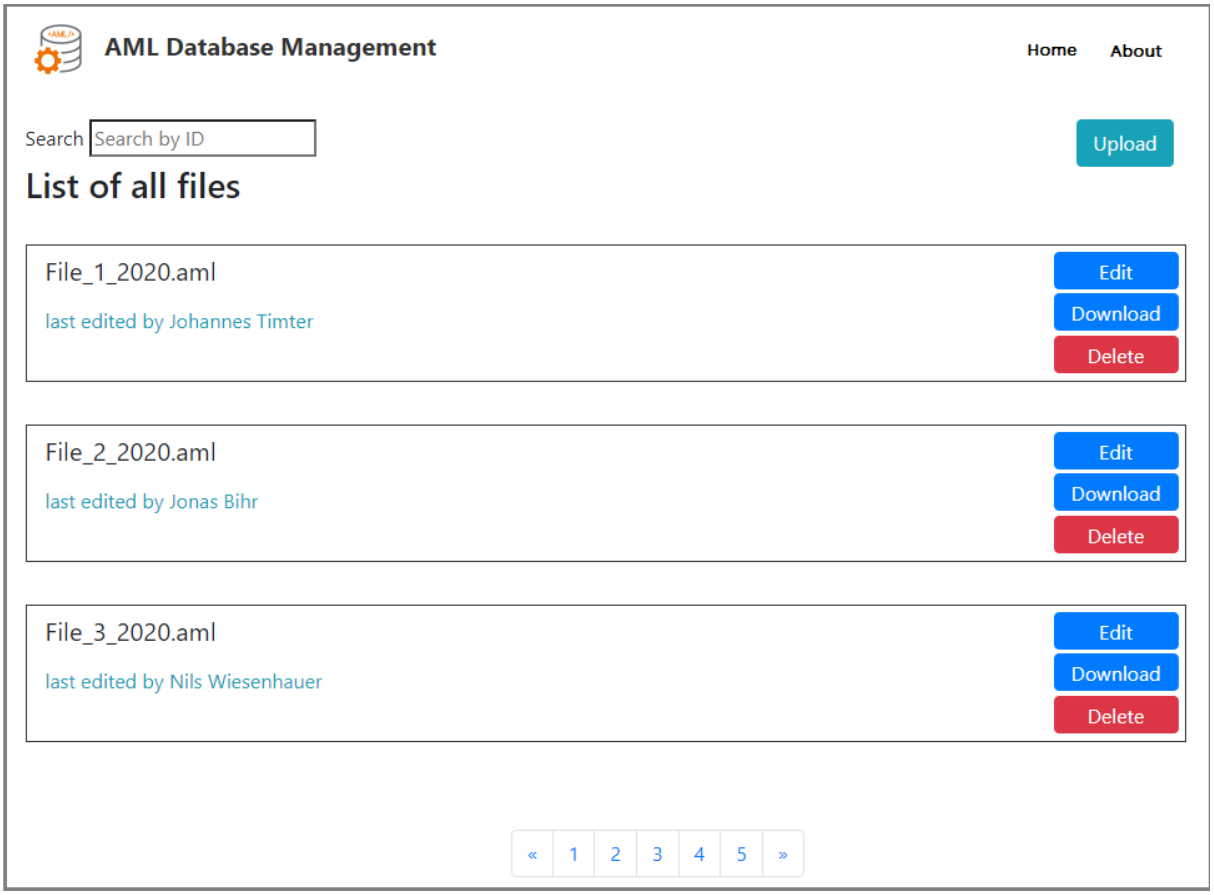

Figure 8 – GUI (Wire frame)

2.2. /LF20/Upload files

The user should have the ability to upload files that are saved locally on the computer of the user with the help of a browser and the graphical user interface. The system should be able to detect if the chosen file is an .aml file and only allow to upload AML files. If the selected file is an .aml file, the file should be uploaded to the server and otherwise the user should see an error message.

Input field	Value Range
File	File to be uploaded

Figure 9 - Upload Button

2.3. /LF30/Conversion from XML to JSON

When the file upload was successful and is now temporarily saved on the server, the system should convert the file to a JSON format, so it can be saved in a MongoDB database. The file should be converted by writing one JSON object with the file content as a string. If this was successful the user sees a success message that the file was saved. When an error occurs during the conversion process, the user is shown an error.

Input field	Value Range
File	File that is temporarily saved on the server

2.4. /LF40/Search for files

The user can search in the database for existing entries using the ID of the file. He can type the ID in a search field on the web page and the application searches in the database for matching entries. The entries that are found, are shown in the user interface in a list of files. He can then select each file and click on it. He also has the ability to filter elements, for example by name or upload date.

Input field	Value Range
Search term	The searched term, used to filter the entries

Figure 10 - Search field

2.5. /LF50/Download files

Once the user has uploaded some files and the files are saved to the database, the user should also be able to download the saved files from the database. The user selects a file that he wants to download and then clicks on a download button. The saved file is then converted back into an AML file and the download from the server starts by beginning the download process from the browser.

Input field	Value Range
File	File to be downloaded

Figure 11 - Download Button

2.6. /LF60/Edit files

The user has the ability to edit saved files in the database. He selects a file to be edited and then presses an edit button. The file is converted to text and is shown to the user in the web interface, where he can edit the file. When the user is done with editing, he can save the changes to the database by clicking on a save button. Alternatively, the user can discard the changes by clicking a close button.

Input field	Value Range
File	File to be edited

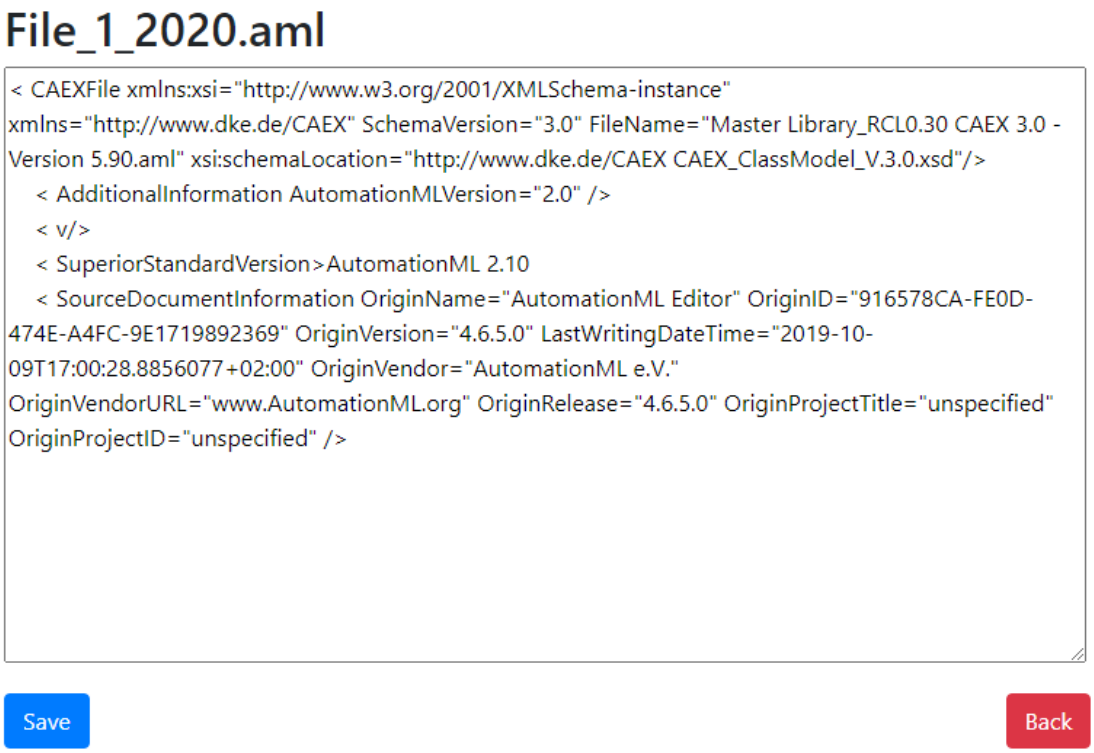

Figure 12 Edit files

2.7. /LF70/Delete files

The user can select a file, saved in the database and then click on a delete button, to delete a saved file. When the button is clicked, a confirmation dialog is opened where the user can choose to confirm or cancel the deletion. On confirmation, the system deletes the database record and the user is shown a success message on a successful deletion.

Input field	Value Range
File	File to be deleted

Figure 13 - Delete Button

2.8. /LF80/List all files

The software shall be able to provide the user with the ability to list all files that are saved in the database. In the web-interface, all the entries are listed in boxes with buttons on them, which provides access to the functions listed above. When there are too many entries available for one page, the system provides a pagination system, which allows a more user-friendly listing of the files.

Figure 14 - List of files

2.9. /LF90/Info Page

The user has the ability to click on the info page to see how the page works and more information on the development. On the info page the web-interface is explained so that the user has a guide when he does not know how a function works.

2.10. /LF100/Database

When a file is uploaded to the server it is converted from an AML format to a JSON format so that it can be saved in the MongoDB database. In the database all the files are stored in an efficient way. The content of the files is saved as a string and then converted into one JSON object, so that each file has their own database entry, with the JSON object as the content.

2.11. /LF110/REST API

The system should support an interface which supports the communication between the frontend and the database. This is achieved with the help of the backend based on NodeJS and ExpressJS. Here the necessary requests are defined and the connection to the database is established. The GUI sends requests to the backend where they are processed. For example, when the user uploads a file, he does this with the help of the GUI. The uploaded file is then sent to the backend where it is converted and stored in the database.

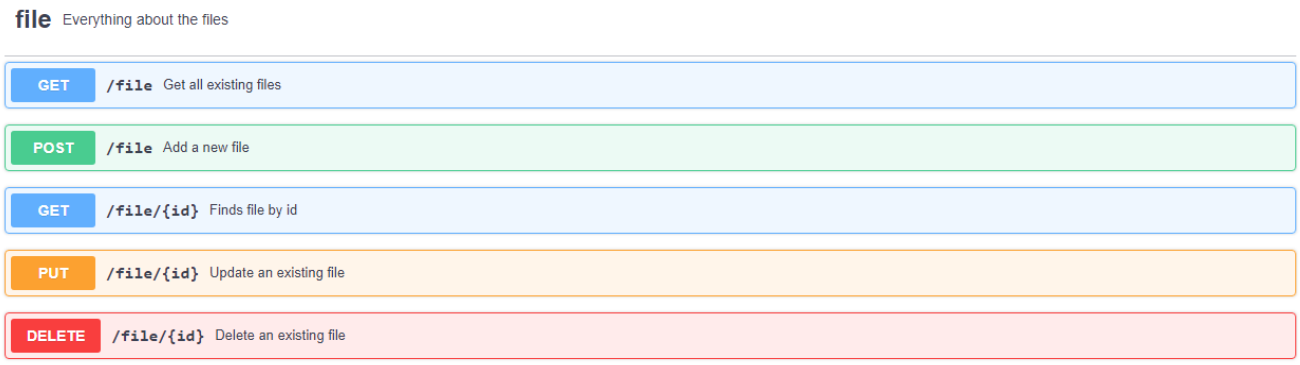

Figure 15 - REST API Swagger

3. Product Data

This section describes the various data the application uses.

3.1. /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.

3.2. /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.

3.3. /LD30/NPM Packages

The REST API is developed based on NodeJS. NodeJS supports packages that can be installed and then used in the project. This application has the package ngx-translate installed which is for automatically translating text on the GUI, to the selected language. The package express.js is also installed to support the development of the backend. Lastly the package JSend is installed to assist with the formatting of the JSON responses of the REST API.

4. Non-Functional Requirements

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

4.1. /NF10/Usability

The system shall support a graphical user interface, which is intuitive, easy to use and accessible from a web-browser.

4.2. /NF20/Expandability

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.

4.3. /NF30/Data Integrity

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

4.4. /NF40/Availability

The system should run without any downtimes and be accessible from any location at any time.

4.5. /NF50/Internationality

The application should be available in several languages, both German and English will be supported.

4.6. /NF60/System Environment

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

5. Figures

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

7. Glossar

AML Automation Markup Language is an open standard data format for storing and exchanging plant planning data.

Angular Angular is a TypeScript based front-end framework which is published as open source software ExpressJS

ExpressJS is the most popular Node web framework and is the underlying library for several other popular Node web frameworks. It provides many mechanisms.

GUI Graphical User Interface

MongoDB MongoDB is a document-oriented NoSQL database used for high volume data storage.

NodeJS NodeJS is a JavaScript free and open source cross-platform for server-side programming that allows users to build network applications quickly.

XML Extensible Markup Language is a markup language to save data in an organized way, to make it human- and machine-readable.

REST Representational State Transfer

API Application Programming Interface