**Software Architecture Specification**

**SAS**

By Lara Lorke

**Table of contents**

1. [Introduction](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#1-introduction)
2. [Scope](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#2-scope)
3. [System Overview](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#3-system-overview)
   * [System Environment](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#31-system-environment)
   * [Software Environment](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#32-software-environment)
4. [Architecture](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#4-architecture)
   * [Architecture Origin](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#41-architecture-origin)
   * [Architectural Concept](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#42-architectural-concept)
   * [Architecture Model](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#43-architecture-model)
5. [System Design](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#5-system-design)
   * [User Interface](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#user-interface)
   * [Data Processing](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#data-processing)
   * [Communication](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#communication)
6. [Interfaces](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#6-interfaces)
   * [User Interface](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#61-user-interface)
   * [Backend Interface](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#62-backend-interface)
7. [Data Flow](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#7-data-flow)
   * [Standard Activities](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#71-standard-activities)
   * [Accessing the Nameplate Generator](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#72-accessing-the-nameplate-generator)
   * [Error Handling of invalid URLs](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#73-error-handling-of-invalid-urls)
8. [Subsystem Specification](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#8-subsystem-specification)
   * [Server Connection](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#81-server-connection)
   * [Display Assets](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#82-display-assets)
   * [Filter&Search](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#83-filtersearch)
   * [Errorhandling](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#84-errorhandling)
   * [Support of Bill of Material](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#85-support-of-bill-of-material)
   * [Support of Namplate Generator](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#86-support-of-namplate-generator)
9. [Modules](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#9-modules)
   * [MOD01: Server connection](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#mod01-server-connection)
   * [MOD02: Display Assets](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#mod02-display-assets)
   * [MOD03: Filter&Search](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#mod03-filtersearch)
   * [MOD04: Errorhandling](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#mod04-errorhandling)
   * [MOD05: Support of Bill of Material](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#mod05-support-of-bill-of-material)
   * [MOD06: Support of Nameplate Generator](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#mod06-support-of-nameplate-generator)
10. [Technical Concept](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#10-technical-concept)
    * [Deployment](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#101-deployment)
    * [Modularization](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#102-modularization)
    * [Data Validation](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#103-data-validation)
    * [Exception handling](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#142-exception-handling)

**1. Introduction**

This document describes the architecture of the AAS web client, a web application for managing assets and generating digital nameplates according to DIN standards. The architecture was developed to meet the quality assurance attributes of maintainability, performance, compatibility, and reliability.

**2. Scope**

The Software Architecture Specification (SAS) for the AAS web client provides a detailed description of the architecture of the web client. Here are some potential benefits and meanings that can be derived from the document:

* Architectural Structure: The SAS describes the basic structure of the AAS web client, including its components, modules, and their interactions. This provides a clear understanding of how the software is structured and how the various parts are interconnected.
* Dependencies and Interfaces: The specification includes information about external systems, libraries, or services on which the AAS web client depends, as well as the interfaces used for communication between different components. This provides a clear understanding of how the web client interacts with other systems.
* Development and Maintenance: Developers can use the SAS to understand and develop or maintain the code of the AAS web client accordingly. The clear architectural description makes it easier for developers to get involved in the project and make changes without compromising the overall structure.

In summary, the Software Architecture Specification (SAS) provides a comprehensive insight into the architecture of the AAS web client, development and maintenance.

**3. System Overview**

**3.1 System Environment**

The application is a web application that runs locally on the user's computer. The client can be accessed via browser. A connection to an AAS server is needed to access AAS objects. These objects are retrieved using REST.

**3.2 Software Environment**

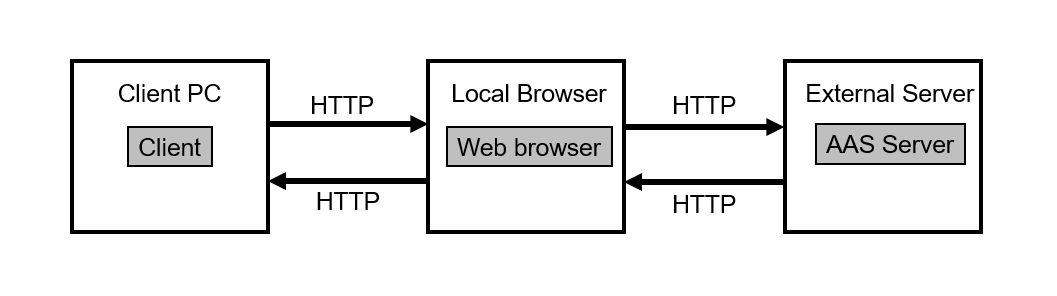
The AAS web client is a client-side single-page application developed with React.js and utilizes JavaScript for logic and CSS for styling. Additionally, the Bootstrap framework was used for further styling. The application communicates via HTTP(S) with a RESTful API backend, which retrieves the required data from an external AAS server.

**4. Architecture**

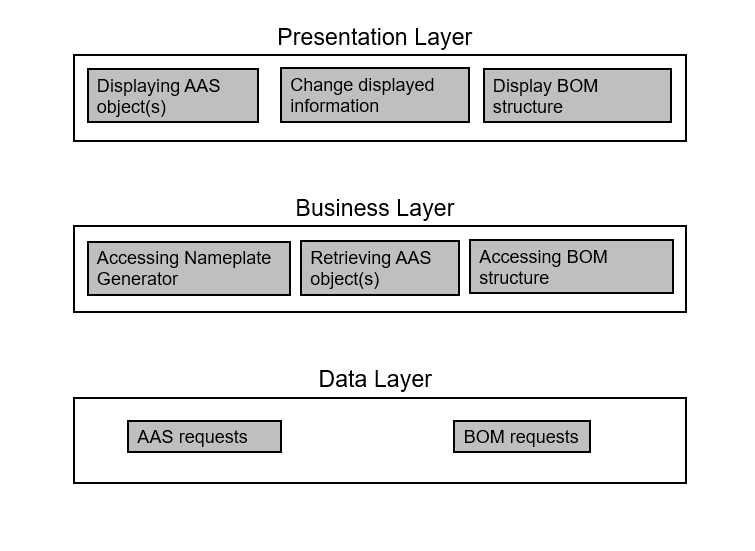
**4.1 Architecture Origin**

The original architecture can be found here [Link](https://github.com/mercal03/TINF21C_Team1_AAS-Server-Webclient/wiki/System-Architecture-Specification-(SAS)). In addition to the provided architecture is expanded by the following points.

**4.2 Architectural Concept**

For the additional implementation the following concept will be used which is based on the previous project. Therefore the concept decision can be found there: [Link](https://github.com/mercal03/TINF21C_Team1_AAS-Server-Webclient/wiki/System-Architecture-Specification-(SAS)) 

**4.3 Architecture Model**

The layered architecture diagram illustrates the logical structure, describing the technical processes based on system objectives and user requirements. The presentation layer manages interactions between users and the software. The business layer houses objects responsible for executing business processes, reflecting the project's objectives. To obtain the necessary information for the business layer, the persistence layer contains AAS-Server requests to retrieve AAS data from an external AAS Server. 

**5. System Design**

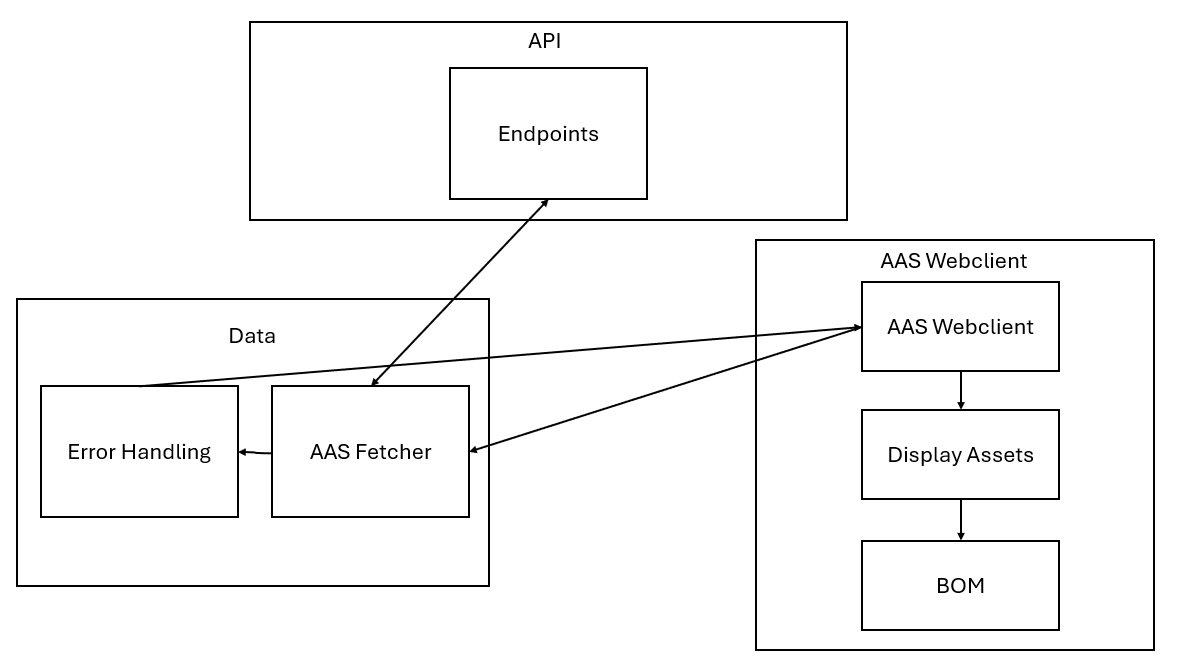
The system design encompasses the following aspects:

User Interface: The user interface is intuitively designed, allowing users to interact with the application. (Partially implemented)

Data Processing: The application processes data entered by users as well as data retrieved from external AAS servers to generate digital nameplates. (Implementation required)

Communication: The application communicates via HTTP(S) with an external AAS server to retrieve and store asset data. (Partially implemented)

Model 5.1 below showcases the technical architecture, delineating the realization of the logical architecture. It serves as a blueprint for implementing the system, detailing its components, classes, and data storage. The diagram, presented as a class diagram, illustrates the classes within the system and their interconnections. Certain components, which serve solely a design purpose, are excluded from the diagram to maintain clarity. Following the diagram, a table offers the class names alongside brief descriptions of their functionalities, augmenting the comprehension of the diagram.



**6. Interfaces**

**6.1 User Interface**

The user interface of the AAS web client is accessible via a web browser, allowing users to interact with the application. The frontend is developed with HTML, JavaScript, React and BootStrap.

**6.2 Backend Interface**

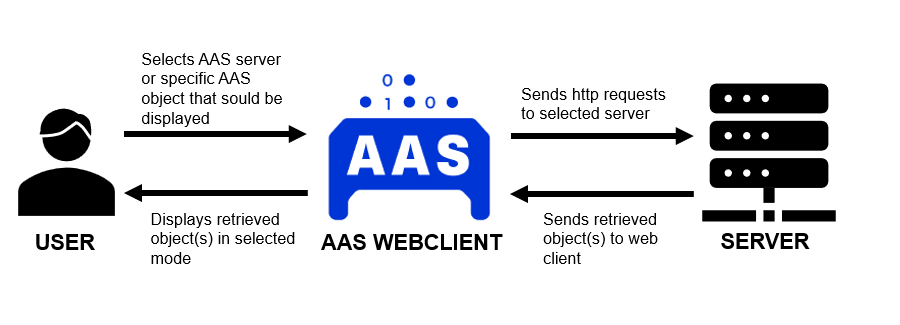
The backend of the AAS web client provides a RESTful API through which users can interact with the system. The backend is implemented with JavaScript. Data is stored in a JSON file.

**7. Data Flow**

**7.1 Standard Activities**

The data flow for standard activities in the AAS web client includes the following steps:

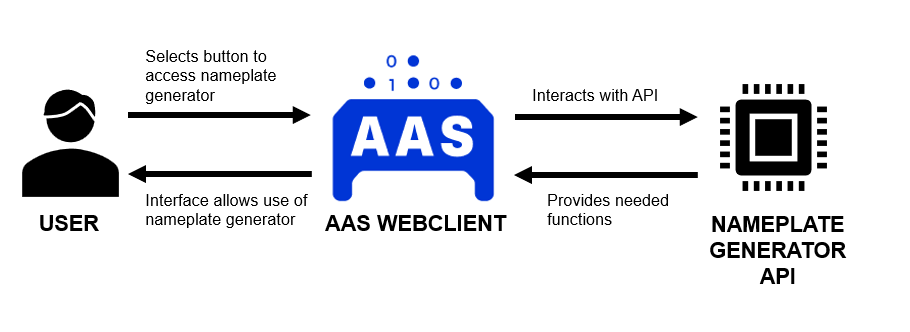
1. The user accesses the user interface via a web browser.
2. The user interface provides a search bar and a server selection to interact with the backend through the RESTful API to retrieve and store data.
3. The backend communicates with external AAS servers to retrieve asset data.
4. The user interface is updated to display the retrieved asset data and allow the user to select assets.



**7.2 Accessing the Nameplate Generator**

The data flow for access to the nameplate generator includes the following steps:

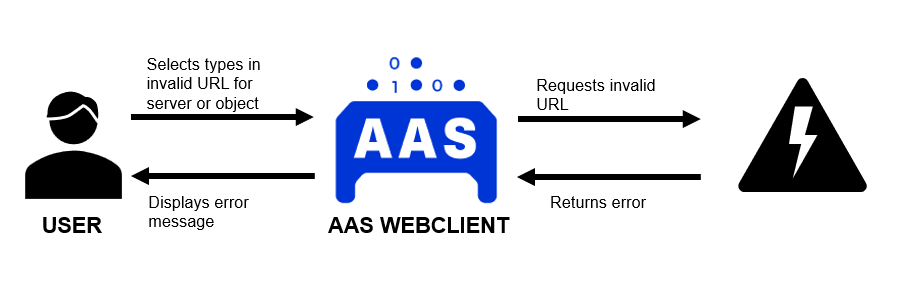
1. The user accesses the user interface via a web browser.
2. The user interface displays a button to interact with the backend through the RESTful API to receive access to the nameplate generator.
3. The backend communicates with the external nameplate generator API to provide its functions.
4. The user interface is updated to display the nameplate generator.



**7.3 Error Handling of invalid URLs**

The data flow for invalid URLs (Server or specific AAS object) includes flowing steps:

1. The user accesses the user interface via a web browser and types in an invalid URL.
2. The user interfaces communicates the invalid URL to the backend.
3. The backend requests invalid URL from server / requests invalid server URL.
4. The backend receives error or awaits 10 seconds request time.
5. The backend returns error to user interface.
6. The user interface displays error message.



**8. Subsystem Specification**

**8.1 Server Connection**

| **Subsystem specification ID** | **MOD01** |
| --- | --- |
| System requirements covered | FR01, FR02 |
| Service | This module is responsible for creating a connection to a AAS server. |
| Module documentation | [MOD01 short](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#mod01-server-connection) [MOD01 long] (<https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/blob/main/Documentation/MOD/TINF22F_MOD01_Team_1_AAS-Webclient.pdf>) |

**8.2 Display Assets**

| **Subsystem specification ID** | **MOD02** |
| --- | --- |
| System requirements covered | FR01, FR02 |
| Service | The module "Display Assets" is responsible for the rendering and presentation of assets and their associated data obtained from a JSON file provided by the AAS server. |
| Module documentation | [MOD02 short](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#mod02-display-assets) [MOD02 long](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/blob/main/Documentation/MOD/TINF22F_MOD02_Team_1_AAS-Webclient.pdf) |

**8.3 Filter&Search&Download**

| **Subsystem specification ID** | **MOD03** |
| --- | --- |
| System requirements covered | FR01 |
| Service | The module "Filter & Search" is responsible for searching for specific AAS objects with the search bar and filtering objects based on year or manufacture date. Based on the already existing search module, further implementation is required to support the search of specific AAS objects via a specific link and download the object. |
| Module documentation | [MOD03 short](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#mod03-filter&search) [MOD03 long](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/blob/main/Documentation/MOD/TINF22F_MOD03_Team_1_AAS-Webclient.pdf) |

**8.4 Errorhandling**

| **Subsystem specification ID** | **MOD04** |
| --- | --- |
| System requirements covered | NFR03 |
| Service | The module "Errorhandling" is responsible for displaying proper error messages for the user. If an error occurs while using the AAS Web client, this module takes code occurring error messages and makes them visible and understandable for the user |
| Module documentation | [MOD04](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#mod04-errorhandling) |

**8.5 Support of Bill of Material**

| **Subsystem specification ID** | **MOD05** |
| --- | --- |
| System requirements covered | FR04 |
| Service | The module "Support of Bill of Material" is responsible for processing and displaying the Hierarchical Structures enabling Bills of Material concerning a AAS object. |
| Module documentation | [MOD05 short](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#mod05-support-of-bill-of-material) [MOD05 long](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/blob/main/Documentation/MOD/TINF22F_MOD05_Team_1_AAS-Webclient.pdf) |

**8.6 Support of Nameplate Generator**

| **Subsystem specification ID** | **MOD06** |
| --- | --- |
| System requirements covered | FR03 |
| Service | Supports the Nameplate Generator ([Link GitHub](https://github.com/TTRSF/TINF22F-Team2-Nameplate-Generator)) |
| Module documentation | [MOD06](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/wiki/System-Architecture-Specification-(SAS)#mod06-support-of-nameplate-generator) |

**9. Modules**

**MOD01: Server connection**

**Scope**

The module "Server connection" is responsible for retrieving AAS objects from a selected server.

**Implementation**

The module is based on the previous project which can be found here: [Link](https://github.com/mercal03/TINF21C_Team1_AAS-Server-Webclient/wiki/MOD01:-Server-connection#6-Implementation)

No further implementation is required.

**MOD02: Display Assets**

**Scope**

The module "Display Assets" is responsible for the rendering and presentation of assets and their associated data obtained from a JSON file provided by the AAS server.

**Implementation**

The module is based on the previous project which can be found here: [Link](https://github.com/mercal03/TINF21C_Team1_AAS-Server-Webclient/wiki/MOD02-Display-Assets#6-Implementation). In addition it is also responsible for altering the displayed information of an AAS Object.

**Further Implementation**

The default display mode is the "user" mode. The function toggleMode() ([Function in filter.js](https://github.com/DerAlbaner180/TINF22F_Team-1_AAS-Webclient/blob/041b485713668fabd7e9a8b73bd5926985536566/SOURCE/aas-webclient/src/filter.js#L26C5-L46C1)) changes the defined prop ([Prop definition in filter.js](https://github.com/DerAlbaner180/TINF22F_Team-1_AAS-Webclient/blob/041b485713668fabd7e9a8b73bd5926985536566/SOURCE/aas-webclient/src/filter.js#L14C1-L24C6)) from one mode to the other. A switch button is responsible for changing the selected modes ([Button in filter.js](https://github.com/DerAlbaner180/TINF22F_Team-1_AAS-Webclient/blob/041b485713668fabd7e9a8b73bd5926985536566/SOURCE/aas-webclient/src/filter.js#L418C17-L430C19)).

Depending on the selected mode different information is displayed. As the default mode is "user mode" it is only check if the mode has changed to "expert". If this is the case all the available information of an AAS object is displayed. Is this not the case only the most relevant information is displayed.

**MOD03: Filter&Search**

**Scope**

The module "Filter & Search" is responsible for searching for specific AAS objects with the search bar and filtering objects based on year or manufacture date. Based on the already existing search module, further implementation is required to support downloading a specific AAS object

**Implementation**

The module is based on the previous project and can be found here: [Link](https://github.com/mercal03/TINF21C_Team1_AAS-Server-Webclient/wiki/MOD03:-Filter&Search#6-Implementation).

**Further implementation**

A button is available for downloading the AAS object. The format which the object will be available in is JSON. The function "jsonDownload()"([jsonDownload()](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/blob/2dd40d250d08d697c1eec57b487badcd1418dc6b/SOURCE/aas-webclient/src/assetBody.js" \l "L31C5-L50C6)) downloads JSON data from shellBody.hide.assetJSON by converting it into a data.json file, creating a download link in the browser, clicking the link to start the download, and then removing the temporary link and revoking the URL.. The code accesses a variable named shellBody.hide.assetJSON which contains the JSON data to be downloaded.

The JSON data is converted into a Blob. A Blob is a binary large object used for storing data. Here, JSON.stringify is used to convert the JSON data into a string. The option null, 2 formats the JSON string with indentation for better readability. The Blob is created with the MIME type 'application/json'. [Here](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/blob/2dd40d250d08d697c1eec57b487badcd1418dc6b/SOURCE/aas-webclient/src/assetBody.js#L32)

A URL for the Blob is created, which can be used in the browser to download the Blob. [Here](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/blob/2dd40d250d08d697c1eec57b487badcd1418dc6b/SOURCE/aas-webclient/src/assetBody.js#L35)

An a element (link) is created, and the Blob URL is set as the href attribute. The download attribute of the link is set to 'data.json', which is the name of the file to be downloaded. The link is then appended to the document and automatically clicked to start the download. [Here](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/blob/2dd40d250d08d697c1eec57b487badcd1418dc6b/SOURCE/aas-webclient/src/assetBody.js#L41-L45)

The link is removed from the document, and the URL for the Blob is revoked to free up memory and avoid potential memory leaks. [Here](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/blob/2dd40d250d08d697c1eec57b487badcd1418dc6b/SOURCE/aas-webclient/src/assetBody.js#L48-L49)

[**MOD04: Errorhandling**](https://github.com/mercal03/TINF21C_Team1_AAS-Server-Webclient/wiki/MOD04:-Errorhandling)

The module "Errorhandling" is responsible for displaying proper error messages for the user. If an error occurs while using the AAS Web client, this module takes code occurring error messages and makes them visible and understandable for the user

**Implementation**

The module is based on the previous project which can be found here: [Link](https://github.com/mercal03/TINF21C_Team1_AAS-Server-Webclient/wiki/MOD01:-Server-connection#6-Implementation)

**Further Implementation**

The displayed errors occur when no assets can be found, an URL is invalid, or a server does not answer a request. Then the module will be active. The react component "Error" is responsible for displaying the information ([Error Component in errorMessage.js](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/blob/99dfa8c38cee38d01f7308747940eb2b93a5bd6a/SOURCE/aas-webclient/src/errorMessage.js#L1C1-L14C22)). This error message will be rendered in the header ([reference to errorMessage.js](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/blob/99dfa8c38cee38d01f7308747940eb2b93a5bd6a/SOURCE/aas-webclient/src/header.js#L11C24-L11C51)) To actually activate the error message an error needs to occur. If this is the case the "error-container" in "header.js" will be overwritten by the "errorMessage.js" component and the error message will be displayed ([example from backend.js if request does not return shells](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/blob/99dfa8c38cee38d01f7308747940eb2b93a5bd6a/SOURCE/aas-webclient/src/backend.js#L107C9-L117C10))

**MOD05: Support of Bill of Material**

The module "Support of Bill of Material" is responsible for processing and displaying the Hierarchical Structures enabling Bills of Material concerning a AAS object.

**Implementation**

The function "getSubmodel(value)" ([getSubmodel(value) in assetBody.js](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/blob/76c48c4c81b5e97866180ca4db7ff99ee2aeb62a/SOURCE/aas-webclient/src/assetBody.js" \l "L119-L173)) in the assetBody receives the asset's submodelid and creates an URL witch consist of the chosen server URL +"/submodels"+ the submodelid in base64. The passed value is the already displayed submodelId value from the HTML ([value for getSubmodel(value)](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/blob/76c48c4c81b5e97866180ca4db7ff99ee2aeb62a/SOURCE/aas-webclient/src/assetBody.js#L105)) With a fetch the data is retrieved. After that the HTML paragraph for the submodule is located and filled with the fitting idShort of the submodule. If the idShort is a Bill of Material, for each submodule element a paragraph is created and filled with the elements idShort.

([Füllen des HTML Elements](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/blob/76c48c4c81b5e97866180ca4db7ff99ee2aeb62a/SOURCE/aas-webclient/src/assetBody.js" \l "L162-L167))

**MOD06: Support of Nameplate Generator**

The module "Support of Nameplate Generator" is responsible for accessing the nameplate generator which can be found here: ([Nameplate Generator](https://github.com/TTRSF/TINF22F-Team2-Nameplate-Generator))

**Further Implementation**

The "Nameplate Generator" provides a Rest API to create a nameplate for an asset. To access the nameplate of an asset simply an Api call is needed. Therefore, the nameplate generator must run on a (local) server. To make an Api call the server which provides the asset and the assetID of the asset is needed. The URL lokes like this: *server where namplate is running* + "/NameplateGenerateByReference?" + *server from asset* + "/?shells/ + *asset* [Here](https://github.com/ilire0/TINF22F_Team-1_AAS-Webclient/blob/2dd40d250d08d697c1eec57b487badcd1418dc6b/SOURCE/aas-webclient/src/assetBody.js#L51C5-L62C6)

**10. Technical Concept**

**10.1 Deployment**

The compiled project will be hosted on a http(s) server, providing a JAVASCRIPT Runtime Environment.

**10.2 Modularization**

JavaScript enables an application to be segmented into multiple modules. This should be used to partition the application into independent modules with distinct purposes. The modules should be loosely coupled, allowing them to be replaced without affecting other modules. Circular dependencies must be addressed as outlined in the interface chapter. To ensure modularization and loose coupling, modules must conceal their internal details and only offer general methods to other modules.

**10.3 Data validation**

To retrieve data a fetch is send. If this fetch fails, the user should be informed about the reason this error occurred.

**10.4 Exception handling**

In order to avoid errors stemming from user input, we validate the input before proceeding with processing. Should an error arise, such as the AAS server failing to respond or the requested data containing inaccuracies, the user will promptly be informed of the issue. For this the module MOD04 is responsible.