07.05.2024

Customer Requirements Specification

TINF22F, Software-Engineering I,

Ümmühan Ay, 7060837

Project: AAS Webclient

Customer: Markus Rentschler, Ivan Bogicevic

Rotebühlstraße 133, 70197 Stuttgart

Product Manager: David Bauer *(inf22167@lehre.dhbw-stuttgart.de)*

System Architect: Lara Lorke *(inf22132@lehre.dhbw-stuttgart.de)*

System Architect: Armin Taktar *(inf22148@lehre.dhbw-stuttgart.de)*

Documentation: Ümmühan Ay *(inf22124@lehre.dhbw-stuttgart.de)*

Developer: Rafael Sancho Pernas *(inf22078@lehre.dhbw-stuttgart.de)*

Test Manager: Kyle Zieher *(inf22210@lehre.dhbw-stuttgart.de)*

Topics

Goals of the project

The goal of the project is the further development of an AAS web client. There was already a predecessor group working on the project. The predecessor group provided the main features of the project such as frontend and backend, but the client still has some shortcomings that need to be addressed. Among other things, the application should be made more user-friendly by allowing support for server browsing and URL direct calls to individual AAS. In addition, it must be possible to see only the relevant information in user mode. Furthermore, the nameplate generator needs to be fixed and the submodel "Hierarchical Structures enabling Bills of Material" should be supported in the application. There should be a structured online user documentation (using mkDocs). The application should be tested against a diverse AAS server infrastructure. Lastly, the application should be set up in a user-friendly manner on the host system.

Product Environment

The AASPortal (Asset Administation Shell) is a model of the so-called digital twin and is so far the most important tool for Industry 4.0. A digital twin is the representation of a mostly physical object in reality. The image must match the physical object in terms of external characteristics. The predecessor project is based on a project of the Fraunhofer Institute: (Link to the Repository: https://github.com/FraunhoferIOSB/AASPortal).

At the beginning of the project, it was discovered that the server was not working, so we wrote to the Fraunhofer Institute on GitHub. We had to decide whether to work with the previous group's project, or to work with the AAS-project of another further studigroup. We decided us for the projects of the previous students.

Product Usage

3.1 Features

<Feature 1>

The application has some lacks in the following categories:

<Feature 1.1> Clear Error Messages

Errors are not clear in their description for the user and its UI is not modern. It should be clear, that the user sees a clear error message with modern design.

<Feature 1.2> Error Cases are missing

Not every Error Case is catched or defined. For example, if a server-url is not available, the site is loading forever without giving any further feedback. The relevant error cases have to be defined and the errors should be structured in a way to help the user to know what went wrong.

<Feature 2> Integration of the Nameplate-Generator

<Feature 2.1> Implementation itself

The Nameplate-Generator should be successfully implemented in the AAS-Web client project. This means, that when an asset is selected, the QR is Immediately displayed. When the QR-Code is scanned it must return information about the asset.

For more information about the Nameplate-generator, follow this link:

https://github.com/mk28/TINF21C_Team2_AAS_digital_nameplate

<Feature 2.2> Documentation

Theimanual must be updated about the Nameplate-generator, which must be defined in a chapter. Here can you find the repository for the nameplate-generator: https://github.com/TTRSF/TINF22F-Team2-Nameplate-Generator

<Feature 3> Support of the Submodule >Hierarchical Structures enabling Bills of Material”

Every asset has a structure in its building. The first object is the asset itself (for example a car). When an user clicks on it, a brighter table should be opened, where the parts of the object can be clicked on (wheel, back mirror, and so on). When the user clicks on the parted pieces of the objects he can see the scope of materials, which are needed for each part of the object.

For more information, follow this link:

https://github.com/admin-shell-io/submodel-templates/tree/main/published/Hierarchical%20Structures%20enabling%20Bills%20of%20Material/1/0

3.2 Use Cases

<Use Case 1> Project Hosting on GitHub Pages

The user can use the project by clicking on a link in GitHub.

<Use Case 2> User-friendly installation of the Web application to the Host-System

The application can be installed in an uncomplicated way to the Host-system.

<Use Case 3> Improves Usability

<Use Case 3.1> User Mode

The user wants a user mode, where he only sees the relevant information of the assets, such as the manufacturer, the name, the id and the nameplate-generator. It should be like that that the default user sees a clear, structured design of the page and that the user knows here to find this information.

<Use Case 3.2> Expert Mode

The user wants an expert mode, where he sees all the information about the asset, also the materials and the URL for the asset.

<Use case 4> Support of Server Browsing and direct URL-Call

<Use Case 4.1> Server Browsing via the table of contents

The user wants to use a table of contents. When the user clicks on one chapter, he will be redirected to this page.

<Use Case 4.2> Calling asset by URL

The user wants to call the asset by an URL. The whole URL should be only available in the expert mode. The URL itself shall the response the JSON Formation of the assets.