Software Requirements Specification

AAS Digital Nameplate Generator

Customer:	Rentschler & Holder
Company address:	Rotebühlplatz 41, 70178 Stuttgart
Supplier:	Team 2

Role	Name	Email Address
Team Lead	Adrian Khairi	Inf21196@lehre.dhbw-stuttgart.de
Product Manager	Sophie Kirschner	Inf21083 @lehre.dhbw-stuttgart.de
Test Manager	Janin Ahlemeyer	Inf21006@lehre.dhbw-stuttgart.de
System Architect	Mika Kuge	Inf21059@lehre.dhbw-stuttgart.de
Technical Documentation	Maris Koch	Inf21050 @lehre.dhbw-stuttgart.de
Software Developer	Erika Zhang	Inf21174@lehre.dhbw-stuttgart.de

Version Control

Version	Date	Author	Comment
1.0	04.10.2022	Janin Ahlemeyer	Initialized the SRS and created a first version
1.1	06.10.2022	Janin Ahlemeyer	Fixed inaccuracies, e.g., diagrams and refined the formulation with Adrian Khairi’s - and Erika Zhang’s review
1.2	09.10.2022	Janin Ahlemeyer	Corrected the document with Adrian Khairi’s, Mika Kuge’s, Erika Zhang’s and the customer’s comments
1.3	14.10.2022	Janin Ahlemeyer	Refined the document with Adrian Khairi’s and Erika Zhang’s comments
1.4	22.10.2022	Janin Ahlemeyer	Added chapter prototype with the assistance of Mika Kuge and Maris Koch’s User Manual

Table of Contents

1 Introduction

The objective of this document is to provide a detailed overview of the product by outlining guidelines and specifications. Thus, setting the foundation of the project and creating an agreement for all the stakeholders involved. Furthermore, it should operate as a bridge between product management development. Therefore, helping the technical team to design and develop the software.

2 Scope

The main objective of this project is to create a nameplate generator for an Asset Administration Shell, also known as “AAS”. Furthermore, a user-friendly front-end application shall be designed and implemented utilizing React. This includes a home page where the user can enter a server address. After selecting the server, the user shall be directed to a user interface (UI) listing all the assets available on the server. Additionally, the interface shall display the data regarding the asset chosen by the user in an organized and clear structure. Both search functions contain autocomplete. The interface allows the communication between any AAS server through REST-API as well as the ability to generate QR codes according to the DIN standard. Additionally, there shall be an option to download the data in SVG or PNG format. The application shall be tested to ensure compatibility with a diverse AAS server infrastructure.

The purpose of creating digital nameplates is to make nameplates environment-friendly due to a paper free solution. Moreover, it provides access to customers everywhere since it is not physically bound to a specific place. Not to mention the virtually unlimited space that can be used to display the information in more detail in reference to multiple assets. Furthermore, the information is dynamic. Thus, it can easily be changed and modified if necessary.

The nameplate generator is for customers that not only want to generate and print the type plates but also use it as a modern solution to provide a better service for their company as well as their own customers. It guarantees a simpler information collection process as well as unlimited access from any place.

3 Overall Description

3.1 Product Perspective

As visible in figure 2.1, the front-end application is a React based project built using HyperText Markup Language (HTML), JavaScript (JS) and Cascading Style Sheets (CSS). It shall be deployed on a server and accessed through the internet with a web browser.

The product relies on the AAS, which provides the information about the assets the product is going to display and utilize. This is accomplished with requests using the REST-API.

The user's web browser acts as the execution environment loading the application through Hypertext Transfer Protocol (Secure), also known as HTTP(S), as well as acquiring the data through REST-API from the AAS-Server.

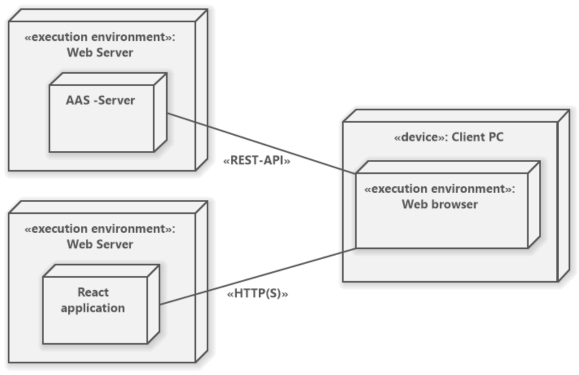
«execution environment»: Web Server
AAS -Server
«REST-API»
«device»: Client PC
«execution environment»: Web browser
«execution environment»: Web Server
React application
«HTTP(S)»

Figure 2.1: Client Model Deployment Diagram

3.1.1 User Interfaces

A graphical user interface (GUI) consisting of a home, table and detail page need to be designed and created. It shall be coherent, e.g., using terminology effortlessly understood by the intended users or rather the target group as well as consisting of a dynamic design meaning it shall adapt to different screen sizes, e.g. smart phone and laptop. Users shall be able to search for a server or a certain product in 30 seconds. The user shall easily recognize sections of the GUI with the assistance of visual cues such as arrows, bold fonts and highlighting.

The interface has to be consistent, e.g., the buttons and formulations have to be the same throughout the pages. Additionally, the interface shall be compatible to multiple browsers, e.g., Chrome, Firefox and Edge.

3.1.2 Hardware Interface

Since the application runs using the internet, it requires the aptitude to connect to it. Additionally, it shall support most devices by running on smart phone as well as computers.

3.1.3 Software Interfaces

The system shall use React v18.2.0 and request the data using HTML5 fetch API. The single page front-end application shall be built using HTML, JS and CSS. Source and destination format of data include JSON.

3.1.4 Communication Interface

The communication architecture abides the client-server model thus employing a REST-compliant web service. The system shall use the HTTP(S) for communication over the internet.

4 Use Cases

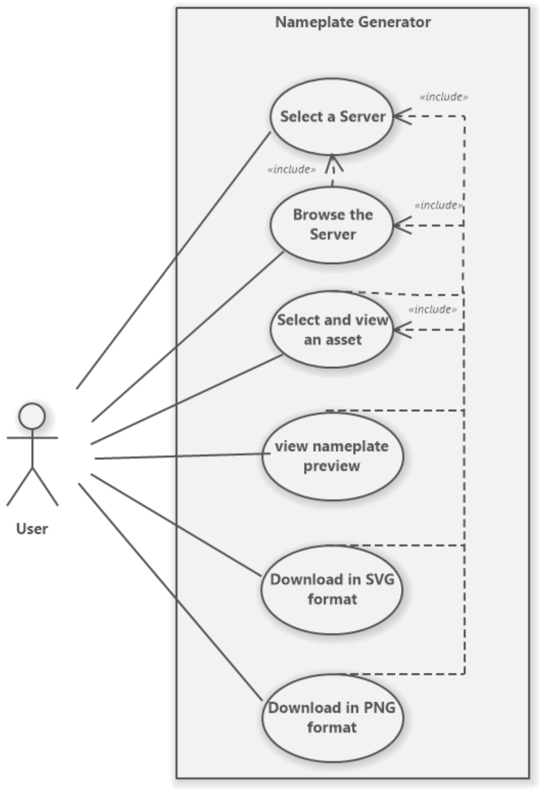

Figure 4.1: Use Case diagram

4.1 UC01 Select a Server

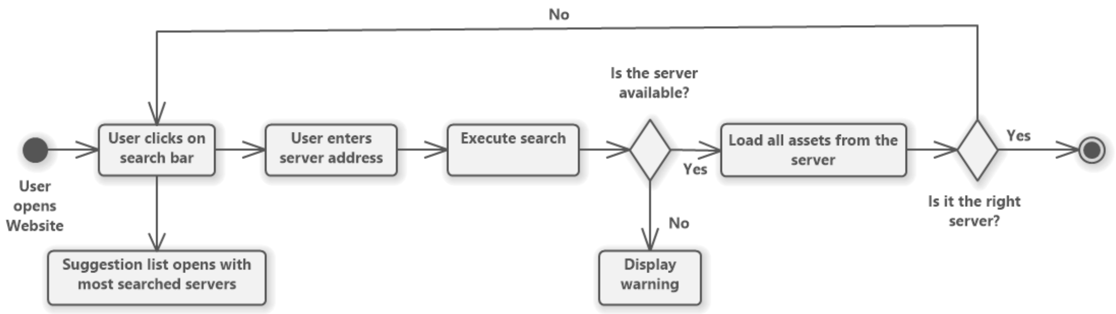
No
Is the server available?
User clicks on search bar
User enters server address
Execute search
Yes
Load all assets from the server
Yes
User opens Website
No
Is it the right server?
Suggestion list opens with most searched servers
Display warning

Figure 4.2: Select a server flow chart

Use Case ID	UC01
Description	The user gains access to a server by entering the server address into the search bar. When clicking on the search bar a suggestion lists opens and shows the most visited servers. The search function contains autocomplete to ensure easier handling.
Involved roles	User, AAS-Server
System boundary	AAS-Server, web browser
Precondition	The user knows the server address and the server must exist as well as being available.
Postcondition on success	The interface visualizes a table filled with all the assets that the server contains.
Triggering event	The user opens the website and wants to view the assets of a specific server.

4.2 UC02 Browse the Server

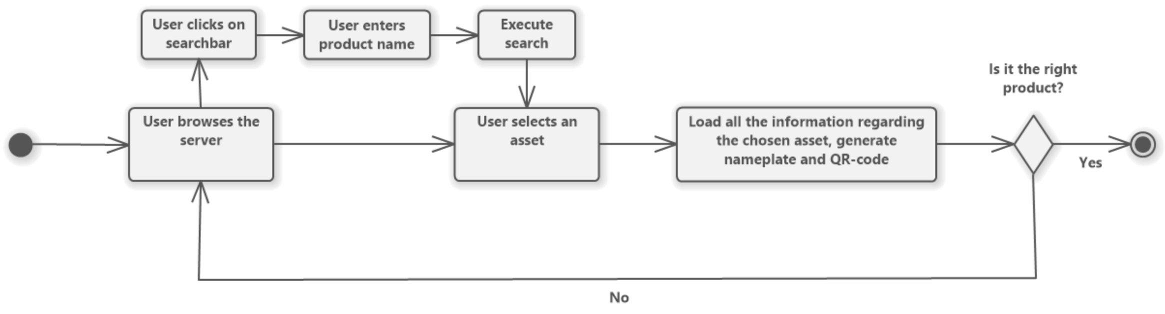

Figure 4.3: Browse the server flow chart

Use Case ID	UC02
Description	Acquire the information of an asset by searching for the product name in the asset list directly or with the search bar. The search function contains autocomplete to ensure easier handling.
Involved roles	User, AAS-Server
System boundary	AAS-Server, web browser
Precondition	The user knows the name of the product and is on the right server containing that specific product.
Postcondition on success	The table displays the product names of the assets that match the search key.
Triggering event	The user wants to find a certain product.

4.3 UC03 Select and view an asset

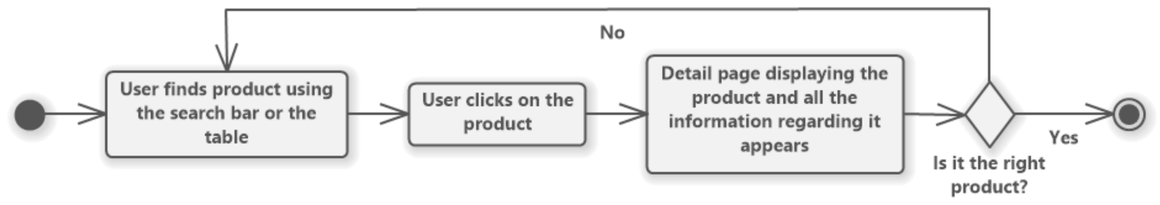
No
User finds product using the search bar or the table
User clicks on the product
Detail page displaying the product and all the information regarding it appears
Yes
Is it the right product?

Figure 4.4: Select and view an asset flow chart

Use Case ID	UC03
Description	It describes the selection of an asset and displaying the data regarding it.
Involved roles	User, AAS-Server
System boundary	AAS-Server, web browser
Precondition	The user is on the right server containing that specific product. Additionally, if there are multiple assets with the same name, the user knows which one they are searching for.
Postcondition on success	The user is lead to a page displaying the data regarding the chosen asset.
Triggering event	The user wants to view the data of a selected product.

4.4 UC04 View nameplate preview

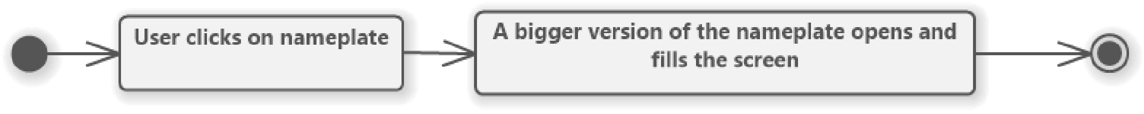
User clicks on nameplate
A bigger version of the nameplate opens and fills the screen

Figure 4.5: Nameplate preview flow chart

Use Case ID	UC04
Description	A preview of the nameplate shall be shown.
Involved roles	User, AAS-Server
System boundary	AAS-Server, web browser
Precondition	A nameplate has been generated.
Postcondition on success	The preview of the nameplate filling the whole screen is displayed.
Triggering event	The user wants to inspect a preview of the nameplate in a bigger version.

4.5 UC05 Download in SVG format

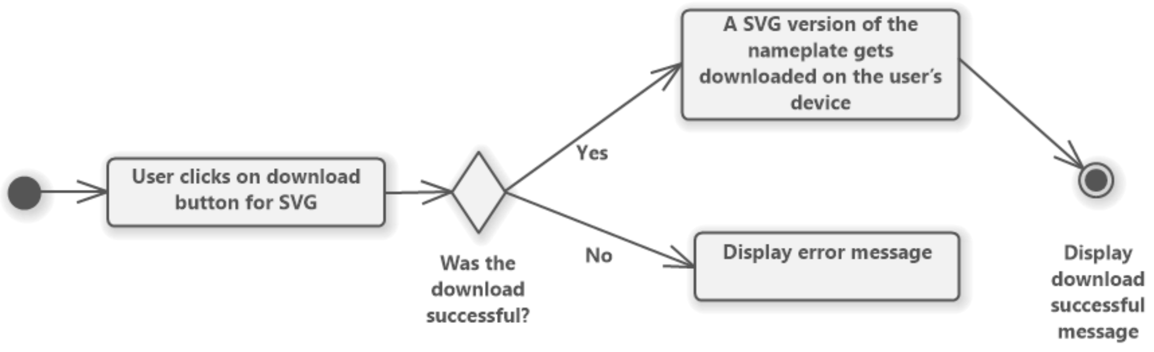
A SVG version of the nameplate gets downloaded on the user's device
User clicks on download button for SVG
Yes
No
Was the download successful?
Display error message
Display download successful message

Figure 4.6: Download in SVG format flow chart

Use Case ID	UC05
Description	The nameplate shall be downloaded in SVG format.
Involved roles	User, User's device
System boundary	Web browser
Precondition	The user's device has enough space to store the file.
Postcondition on success	A SVG version of the nameplate is on the user's device.

Triggering event	The user wants to download the nameplate in a SVG format.

4.6 UC06 Download in PNG format

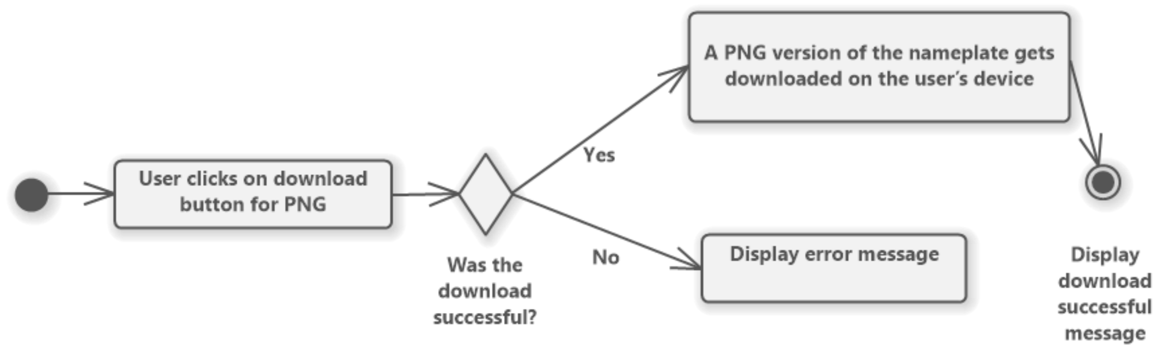
User clicks on download button for PNG
Was the download successful?
Yes
No
A PNG version of the nameplate gets downloaded on the user's device
Display error message
Display download successful message

Figure 4.7: Download in PNG format flow chart

Use Case ID	UC06
Description	The nameplate shall be downloaded in PNG format.
Involved roles	User, User’s device
System boundary	Web browser
Precondition	The user’s device has enough space to store the file.
Postcondition on success	A PNG version of the nameplate is on the user’s device.
Triggering event	The user wants to download the nameplate in a PNG format.

5 System Requirements

The requirements shall be described using a requirement number, an overview describing the requirement, originator, fit criterion and a priority number. The requirements shall be ranked from 0 to 5. 0 being the least important and 5 being the highest priority. This enables the developers to determine which requirements, have a higher priority and therefore need to be dealt with first or which ones are rather optional.

5.1 Functional Requirements

5.1.1 REQ1 Responsive and compatible GUI

Requirement ID	REQ1
Overview	The interface is built on a responsive web design thus it can be accessed on phone and laptop and the view shall adjust according to the user’s device. It shall also be compatible with multiple browsers.
Priority	2
Originator	Customer
Fit Criterion	Testing whether there is a similar design on laptop and phone as well as on different Browsers, for instance Chrome, Firefox and Edge.

5.1.2 REQ2 Dark and light mode menu

Requirement ID	REQ2
Overview	The buttons shall enable switching between light and dark mode.
Priority	0
Originator	Team
Fit Criterion	If the user is using light mode and clicks on the dark mode button the colors of the page shall change to a darker theme. If the user is using dark mode and clicks on the light mode button, the color theme shall change to a lighter one.

5.1.3 REQ3 Download menu for SVG and PNG format

Requirement ID	REQ3
Overview	The nameplates are downloaded onto the user’s device.
Priority	4
Originator	Customer
Fit Criterion	A SVG or PNG version of the exact nameplate displayed on the page shall be on the user’s device and the user is notified about the successful download. The format depends on which button is pressed.

5.1.4 REQ4 Search functionality

Requirement ID	REQ4
Overview	The search functions allow the user to search for a certain product or a server.
Priority	3
Originator	Customer
Fit Criterion	It requires a search bar where the user can type in the server or product name with the assistance of autocomplete. When clicking the search button, the right asset on server shall be displayed for the user.

5.1.5 REQ5 Navigation buttons

Requirement ID	REQ5
Overview	When clicking the back button the user shall be led to the page, he previously viewed, while being directed to the following page when clicking the forward button.
Priority	1
Originator	Customer
Fit Criterion	With the back button the user shall be able to move to the previous page. Navigating to the root shall be possible, e.g., from detail page to table page to home page. The forward button shall forward the user to the next page.

5.1.6 REQ6 QR-code generator

Requirement ID	REQ6
Overview	The application is able to generate QR-codes for the nameplates.
Priority	5
Originator	Customer
Fit Criterion	QR-codes shall be generated for every asset and visible on the detail page. They shall correspond to the DIN Standard and contain information in the following order: General Information, Technical Specification, Certificates and Patents.

5.1.7 REQ7 Nameplate generator

Requirement ID	REQ7
Overview	It can create nameplates for the chosen asset.
Priority	5
Originator	Customer
Fit Criterion	Nameplates according to the DIN standard shall be generated out of the asset the user chose. It shall contain all the necessary information such as general Information, warning signs, certificates and a QR-code. A small version shall be displayed on the detail page.

5.1.8 REQ8 Nameplate preview

Requirement ID	REQ8
Overview	The application provides a preview of the nameplate.
Priority	1
Originator	Team
Fit Criterion	The user can click on the nameplate enabling a bigger version to open and fill the screen so it can be looked at in more detail, e.g., zooming in.

5.1.9 REQ9 Error handling

Requirement ID	REQ9
Overview	The system has an error handling.
Priority	4
Originator	Team
Fit Criterion	When the server is down or does not exist, the user shall be notified.

5.2 Non-functional Requirements

5.2.1 NREQ1 User-friendly

Requirement ID	NREQ1
Overview	A user with no experience with the website shall be able to use it effortlessly.
Priority	5
Originator	Customer
Fit Criterion	An inexperienced user shall be able to navigate through the page in two minutes and use search functions in 30 seconds. If the user knows the right server and product name, it shall take them one minute to get to the detail page of the asset they were trying to look at and 20 seconds to download a SVG or PNG version of the nameplate.

5.2.2 NREQ2 Performance

Requirement ID	NREQ2
Overview	The software should maintain a high performance in terms of how fast a website loads including the time fetching data from the server and displaying it.
Priority	3
Originator	Customer
Fit Criterion	The standard loading time of websites is one to two seconds. However, taken into consideration that it depends on the

	internet connection as well, the pages will load in a duration of well below seven seconds.

5.2.3 NREQ3 Reliability

Requirement ID	NREQ3
Overview	The application needs to be reliable in terms of containing the right information.
Priority	4
Originator	Customer
Fit Criterion	The nameplates and QR-codes have to be generated according to the DIN standard. The information must belong to the chosen asset meaning there shall not be a false exchange of data regarding different assets.

5.2.4 NREQ4 Maintainability

Requirement ID	NREQ4
Overview	The website requires a high maintainability.
Priority	3
Originator	Team
Fit Criterion	Each team member shall be able to read and understand the code as well as knowing how to make changes. Additionally, developers not belonging to the team shall be able to do so as well after reading the code for five hours.

5.2.5 NREQ5 License

Requirement ID	NREQ5
Overview	The product is an open source software thus a license for publishing it is required.
Priority	5
Originator	Team
Fit Criterion	The product is published under the MIT license and it is added to the GitHub.

6 UI sketches

A high fidelity prototype of the front-end application has been designed using the tool Figma and can be found under the following link: Figma prototype.

The current illustration on the homepage is only a placeholder and will therefore be replaced with a nameplate later on.

At the top of the page there is a navigation bar containing a back button, either a sun or moon icon, a home, about and GitHub link. By clicking on the moon and sun icons the user can switch between dark and light mode. Through the back button the user can navigate to the previous page. Furthermore, the home link leads to the home page.

After searching for a server by clicking on the magnifying glass the user is led to an asset list where they can search for an asset. When clicking on the last row the table expands and a scrolling function is activated. This is simply for sketch purposes since the table will be generated dynamically. The first row leads the user to an example page of an asset, however, for show purposes it does not contain all the information nor the right nameplate. Lastly, clicking on the nameplate results in a nameplate preview with a bigger version filling the page.

7 Prototype

This chapter is written based on the User Manual in the GitHub Wiki and serves as a foundation for the implementation of the project as well as the prototype.

As described in the previous chapter a Figma prototype has been created for the UI concept, however, the design is an objective of the fourth semester. Therefore, the proof of concept shall mostly contain the functional aspects.

The application shall be a single page application, thus, the current web page is dynamically rewritten with new data so the user never switches to another page unless they click on a link to an external website.

The home page contains a navigation bar consisting of a back button that enables the user to return to the previous page. This is accomplished by using the "React Router" library. The library writes into the browser history, which allows for navigation using the navigation buttons of the browser as well as buttons in the application itself. Alongside with a dark or light mode button that switches icons depending on the current mode the user is in. By clicking on the button, it toggles between a dark and light color theme. On the right side of the bar there shall be a home button, which directs the user to the start page and an about button, that displays further information about the project including but not limited to license information. Lastly, there is a link to the GitHub project. Furthermore, the start page contains a search bar, where the user can enter a server address. By clicking on the button illustrated as a magnifying glass the application connects to the server with the given address. If the server with the address responds, the user shall be forwarded to a list of assets that the server contains.

The asset list is generated dynamically and shall therefore contain as many rows as assets exist on the server as well as one header row. If the list is too long to fit on the screen completely, a scrolling function is activated to enable the user to browse the list. Additionally, the user can search for the asset name in the search bar above the table. Next to the search bar the server address is be displayed to ensure easier navigation.

If the user clicks on a row, the page shall asynchronously load the information of the chosen asset as well as generate a nameplate with a QR-code. The user can click on the nameplate to view it in full screen. Furthermore, the user can click on the download buttons to download the nameplate in either SVG or PNG format to their device.

8 Bug Fixes

Since only a proof of concept has been created so far there have not been any bugs to fix. Therefore, this chapter will be filled in the fourth semester.