Software Architecture 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	01.10.2022	Mika Kuge	Initialize and create a first draft of the SAS
1.1	13.10.2022	Mika Kuge	Refinement based on Adrian Khairi's review
1.2	17.10.2022	Mika Kuge	Refinement based on Erika Zhang's review
1.3	22.10.2022	Mika Kuge	Adding diagrams
1.4	23.10.2022	Mika Kuge	Refinement based on Adrian Khairi's - and Erika Zhang's review

Table of Contents

1 Introduction

The purpose of this document is to provide a detailed description of the software architecture for the digital nameplate generator by focusing on five key quality attributes: Usability, maintainability, performance, compatibility and reliability. These attributes were selected from the ISO/IEC 25010 quality model based on their importance in the design and development of the application [1].

2 Scope

The digital nameplate generator creates nameplates for assets managed in an Asset Administration Shell, also known as “AAS”. Assets eligible for a nameplate, contain a section of data that describes the most important aspects of the asset including but not limited to information about the manufacturer, serial number, year of production and product markings. An AAS Server can be accessed via a REST-API to obtain the stored data.
A web application shall be created that can fetch data using the REST-API and generate a digital nameplate according to the DIN SPEC 91406 [2]. This application shall consist of a backend that fetches the data and creates the digital nameplate. In addition to that a user-friendly frontend, which displays the generated nameplate as well as all nameplate data that was received from the server, shall be developed. This way a user has the possibility to confirm that their asset contains correct and complete data. Additionally, the user can download the generated nameplate in the SVG or PNG format. The application shall be tested to confirm usability, reliability, maintainability, performance and compatibility.

3 System Overview

3.1 System Environment

The generator shall be implemented as a web application, therefore a web browser capable of running JavaScript is necessary. The browser acts as the execution environment for the website that can be served by a Hypertext Transfer Protocol (Secure), also known as HTTP(S), server. A connection to an AAS Server is also needed, as the generator itself does not contain any asset information but fetches it via the REST-API provided by a separate AAS-Server.

3.2 Software Environment

The frontend shall be developed using the React framework v18.2.0, which is an open-source JavaScript library for building user interfaces (UI). In the release process, the proprietary code for the framework will be translated into HTML, JavaScript and CSS. This ensures that the code can be read by a web browser without the need for special installations for the user. For the translation, *node.js* is used, which is an open-source JavaScript runtime environment.

The backend shall use the HTML5 fetch API to get all the necessary data from the server. The data format used to transfer and further process the data is JSON.

3.3 Quality Attributes

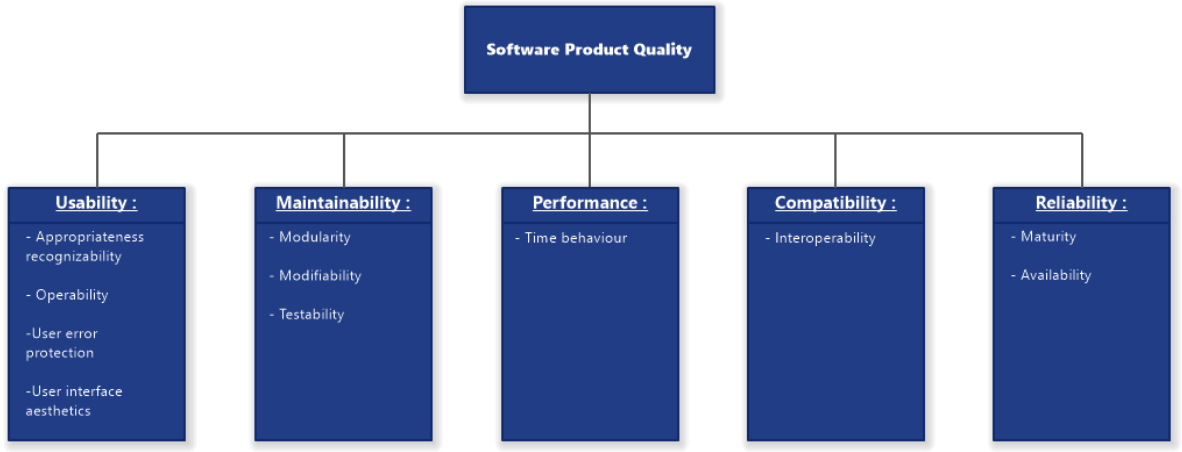

Figure 3.1 Software Product Quality Model

3.3.1 Usability

Usability is one of the most essential attributes since the users can directly see how well it is implemented. The front-end application must fulfil the following criteria to ensure that the user can achieve specified goals with effectiveness and satisfaction.

Appropriateness recognizability: The user can recognize with ease whether the system is appropriate for their demands for instance through an example nameplate on the home page.

Operability: Degree to which the product enables the user to simplify operation and control. The evaluation is focused on buttons and functions facilitating the navigation through the website.

User error protection: The system can handle error states caused by bugs or wrong user inputs and protect itself and the user from them. It can report errors to the user in a precise and clear way, so the user knows what caused the error and eventually how to fix the problem.

User interface aesthetics: This criterion describes how pleasing and satisfying the interaction between the user and UI is. This can be accomplished through a consistent color scheme and design for similar events as well as maintaining UI components in the same position to ensure a recognition value.

3.3.2 Maintainability

This attribute describes how efficiently the software can be modified for its improvement or adaptation.

Modularity: How well the system is broken into discrete compartments to reduce complexity and ensure minimal impacts on other components, when changing one. This aspect can be facilitated by disintegrating the main objective into smaller subgoals, thus separating the system into subsystems.

Modifiability: Describes the degree to which a product or system can be effectively and efficiently modified without causing defects or decreasing the product quality. By creating modules, the code for the subsystems shall be a separate entity. Additionally, the code shall be divided into multiple components and classes hence changing one component or class shall have a minimal effect on other classes and components.

Testability: Describes how well test criteria can be established and applied to determine whether they were met.

3.3.3 Performance

Moving on to the next characteristic which represents the product's performance relative to the amount of resources applied under predefined conditions.

Time behavior: The latency, meaning the time spent responding to an event, for instance the average page load time that should be well below seven seconds.

3.3.4 Compatibility

The quality goal compatibility describes the degree to which the system can exchange information with the AAS-Server and the interaction with different execution environments and hardware. Meaning the web application should run on multiple browsers and devices, for instance, laptop and smartphone.

Interoperability: Defines the degree of two or more systems exchanging information and using said information, e.g. generating nameplates and QR-codes based on the asset data acquired through the REST-API from the AAS-server.

3.3.5 Reliability

Moreover, the criterion reliability defines the ability of the system to continue operating under predefined conditions for a specified period of time.

Maturity: It describes how well the software meets standards and requirements during regular operation.

Availability: This characteristic outlines whether the user can access the application that should be available at any given time.

4 Architecture

4.1 Architectural Concept

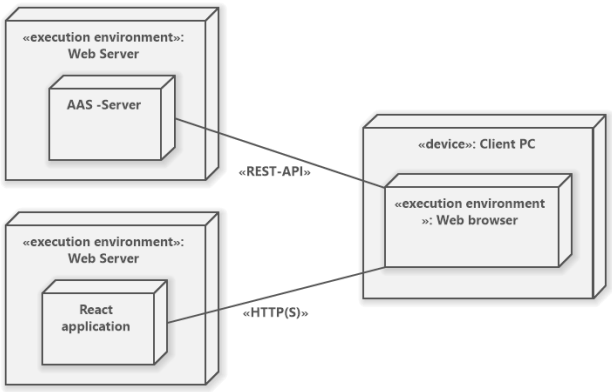

Figure 4.1: Client Model Deployment Diagram

One main objective of the project is to create a front-end application utilizing the JavaScript library React. The UI will be built using Hypertext Markup Language (HTML), JavaScript (JS) and Cascading Style Sheets (CSS).

The web browser of the client shall act as an execution environment and load the application through HTTP(S) from a web server. Asset data will be requested from an AAS-Server using the REST-API provided by the server. All requests will be sent from the browser running the web application.

4.2 Architectural Model

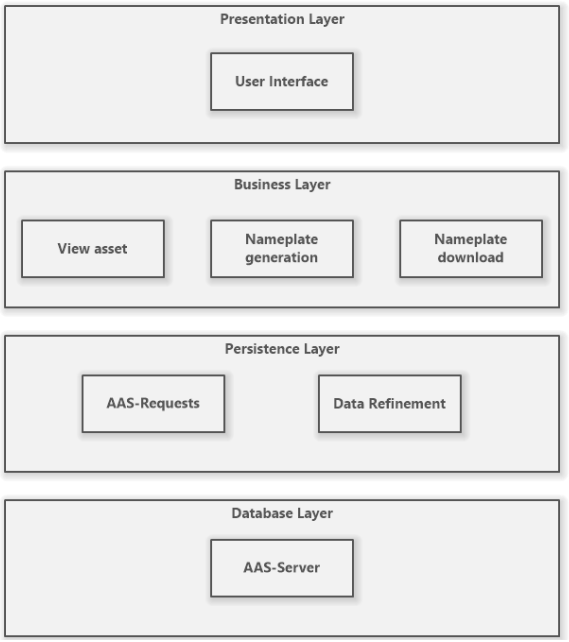

Figure 4.2: Layered Architecture Diagram

The layered architecture diagram illustrates the logical architecture which describe the technical processes based on the system objectives and the user's requirements.

The presentation layer is responsible for any interaction between the user and the software. Thus, it is in charge of displaying information as well as acquiring the user's input and presenting the outcome of processes of further layers. The business layer contains objects that execute the business processes. It is the result of the project's objectives. To get the information needed by the business layer, the

persistence layer containing the AAS-Server requests and the data refinement, must access the Database layer containing the AAS-Server.

5 System design

The following model 5.1 illustrates the technical architecture which outlines the implementation of the logical architecture. It provides a reference model for the implementation of the system in terms of components, classes and data storage.

The figure is a class diagram of the UI's system design. The diagram depicts the classes the system shall contain as well as their connections to each other. Components that just serve a design goal are omitted from this diagram. Additionally, all the classes illustrated in the UI box are for the implementation of the frontend while the classes portrayed outside of it belong to the back end and are responsible for the fetching of the data. Below the figure, a table containing the class name, a short description of its function and the storage location is shown to further explain the diagram.

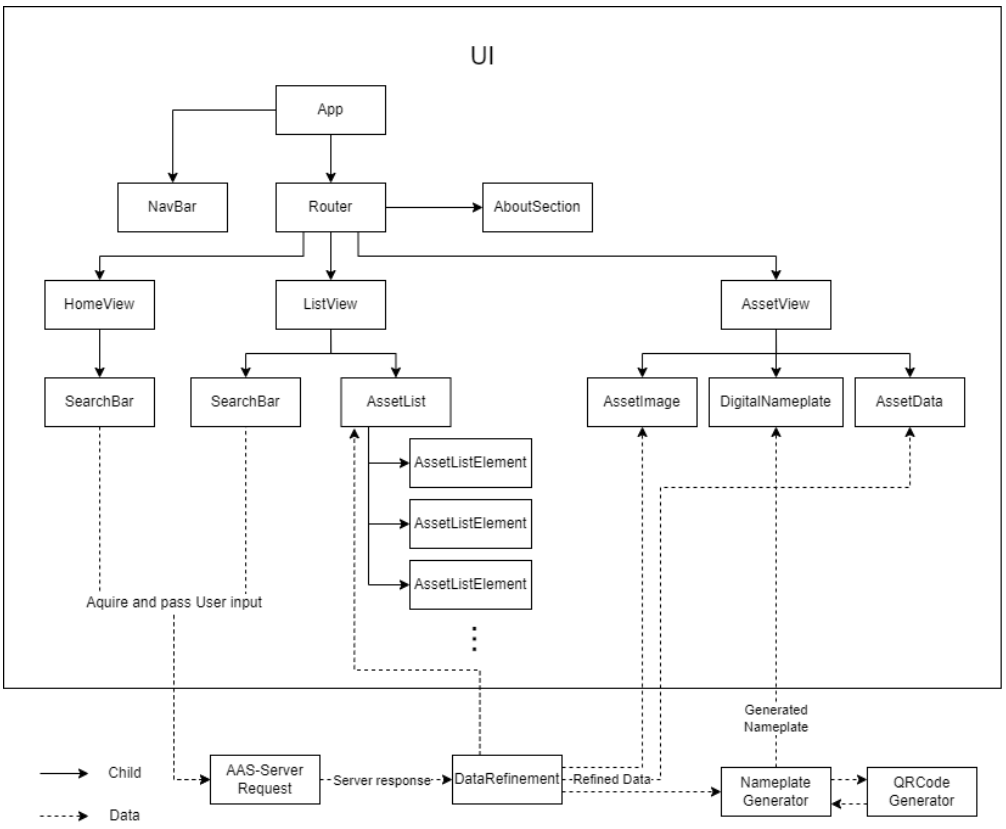

Figure 5.1: UI class diagram

Class name	Description	Covered Requirements and Use Cases
App	The Root of the frontend. It shall contain all further elements of the frontend. Additionally, it shall contain the current state of the application	REQ1 Responsive and compatible GUI
NavBar	The navigation bar shall contain the back button, the dark and light mode menu, the home and about button and lastly, a link to the GitHub project. The navigation bar shall accompany the user on every page.	REQ2 Dark and light mode menu NREQ1 User-friendly
Router	The router shall enable changing the content of the website according to the location in the URL bar of the browser	
AboutSection	This shall contain more information related to the project such as the license information.	NREQ5 License
HomeView	The home view shall contain the start page of the application. It shall be shown as a user opens the website.	UC01 Select a Server
ListView	The list view shall contain the asset list and a search bar to search for an asset by name.	UC02 Browse the Server UC03 Select and view an asset
AssetView	After selecting an asset, the user shall be led to the asset view, which displays all the necessary data about the chosen asset.	UC03 Select and view an asset UC05 Download in SVG format UC06 Download in PNG format

		REQ3 Download menu for SVG and PNG format
SearchBar	The search bar shall acquire and pass the user input to the backend. It shall have an autocomplete function. The server search bar shall display a list of previously visited servers.	REQ4 Search functionality UC01 Select a Server
AssetList	This dynamically generated list shall contain all the assets of the chosen server.	UC02 Browse the Server UC03 Select and view an asset
AssetImage	This shall contain a photo of the selected asset, given that the server contains such an image.	UC03 Select and view an asset
DigitalNameplate	This object shall contain an image of the generated digital nameplate.	REQ8 Nameplate preview
AssetData	This object shall display all data of the chosen asset in text form. It shall also signal if crucial data is missing.	UC03 Select and view an asset NREQ3: Reliability
AssetListElement	This shall contain the name of one asset in the AssetList.	UC03 Select and view an asset
AAS-Server Request	This class shall implement methods to get data from the AAS-Server via the HTML5 fetch-API.	UC01 Select a Server UC02 Browse the Server UC03 Select and view an asset
DataRefinement	The data from the server response shall be refined, filtered, checked for missing crucial data and brought into a standard internal format.	NREQ3 Reliability

NameplateGenerator	This class shall generate the nameplate for a chosen asset.	REQ7 Nameplate generator
QRCodeGenerator	This class shall generate the QR-code needed for generating the nameplate.	REQ6 QR-code generator

6 Product Perspective

6.1.1 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. 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.

6.1.2 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.

6.1.3 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.

6.1.4 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.

7 Subsystem specification

The four main modules are web interface, AAS request, data refinement and nameplate generation. Since there shall be a more detailed version of the module documentation in the GitHub Wiki, this chapter shall only shortly describe the modules. The modules are outlined using a subsystem specification ID, the requirements that are covered by the module as well as the service and interfaces. Additionally, it will contain external data, storage location and module documentation with a link to the GitHub Wiki.

7.1 MOD01 Web interface

Subsystem specification ID	MOD01
System requirements covered	- REQ1 Responsive and compatible GUI - REQ2 Dark and light mode menu - REQ3 Download menu for SVG and PNG format - REQ5 Navigation buttons - REQ8 Nameplate preview - NREQ1 User-friendly
Service	The web interface contains the dark and light mode button that enables the user to toggle between a dark and light color theme. A back button which the user can use to navigate to the previous view. Additionally, it contains a home and an about button. Thus, the user can directly be forwarded to the start page and about page. Further Buttons and input fields allow the user to interact with the page according to the current state of the page.
Interfaces	UI, hardware, software, communication
External data	none
Storage location	Code link
Module documentation	Wiki link

7.2 MOD02 AAS - request

Subsystem specification ID	MOD02
System requirements covered	- REQ4 Search functionality - REQ6 QR-code generator - NREQ3 Reliability
Service	The application must request all asset data form an AAS-Server using its REST-API interface. The different datasets needed for the website to work are implemented in this module.
Interfaces	UI, hardware, software
External data	- Asset list - Asset data
Storage location	Code link
Module documentation	Wiki link

7.3 MOD03 Data refinement

Subsystem specification ID	MOD03
System requirements covered	- NREQ2 Performance - NREQ3 Reliability
Service	On request the server sends more data than necessary to generate a digital nameplate. The raw data from the server must be filtered and checked for completeness and errors to allow further processing in the generation of the nameplate and display of the data.
Interfaces	UI, hardware
External data	Asset data
Storage location	Code link
Module documentation	Wiki link

7.4 MOD04 Nameplate generation

Subsystem specification ID	MOD04
System requirements covered	- REQ6 QR-code generator - REQ7 Nameplate generator
Service	It generates a nameplate corresponding to the DIN SPEC 91406 [2]. It contains general Information, warning signs, certificates and a QR-code.
Interfaces	UI, hardware
External data	Asset data
Storage location	Code link
Module documentation	Wiki link

8 Technical Concept

8.1 Single-page application

The web application shall only contain one HTML page. The content of the page is updated according to the current state of the site and user input. This way the website does not have to be reloaded on input or state change. Data from the server will be loaded into the view as soon as it is available.

8.2 Component-based development

React is a component-based JavaScript framework. This enables the creation of simple components that later form a complex UI. This concept ensures a high maintainability since the components are separated and barely affect each other, thus changes to one component can be made without causing errors to occur in another one. Furthermore, it increases efficiency through the reduction of repetitive code as the components are reusable.

8.3 Deployment

The compiled project shall be hosted on a http(s) server, it does not require an extraordinary infrastructure except providing HTML, CSS and JS to the user.

8.4 Data validation

The data is refined by structuring and filtering it. If a nameplate is generated while information is missing the user is notified about an incomplete nameplate.

8.5 Exception Handling

To prevent errors by user input, user input is validated before it is processed. If an error occurs i.e., because the AAS-Server does not respond or the requested data contains errors, the user will be notified about the problem and if possible given hints on how to fix the problem.

8.6 Asynchronous communication

The web browser shall load the information and generate the QR-code and nameplate asynchronously. This increases perceived performance, the website does not become unresponsive on loading data and the user can view the data while the QR-code and the nameplate are still being generated.

9 Architecture Compliance Review

The key to a successful project lies in the planning and documentation of the project. Thus, the architectural concept was developed as a basis for the implementation of the software. This chapter shall contain a review of the compliance of the Digital Nameplate Generator against the determined architectural concept and quality attributes. However, this review must be conducted in the fourth semester when all of the business requirements are fulfilled and the project has established an IT architecture. Though the analysis shall take place before the end of the finished product to catch inaccuracies along the way and ensure a product surpassing the customer’s expectations.