Table of Contents

- 1 Scope
- 2 Glossary
- 3 Module Requirements
  - 3.1 User View
  - 3.2 Requirements
  - 3.3 Module Context
- 4 Analysis
- 5 Design
  - 5.1 Risk
- 6 Implementation
- 7 Module Tests
  - 7.1 Module Test Plan
  - 7.2 Module Test Report
- 8 Summary

Changelog

Version	Date	Author	Comment
1.0	24.10.2022	Maris Koch	Chapters 1-5
1.1	09.05.2023	Maris Koch	Reworked Chapters 1-5
1.2	09.05.2023	Maris Koch	Added implementation details and updated images
1.3	10.05.2023	Maris Koch	Implemented Feedback by Mika Kuge
1.4	11.05.2023	Janin Ahlemeyer	Added Module Tests

1 Scope

This module provides a graphical user interface (GUI), which is an intuitive illustration of content. It is the frontend of the nameplate generator guides the user through the workflow from entering the server address to downloading the desired nameplate.

2 Glossary

- **GUI** - graphical user interface
- **AAS** - asset administration shell
- **AASX** - file format to store an asset
- **AASX server** - server, that can store AAS assets and has a standardized API specified in the GitHub repository

3 Module Requirements

3.1 User View

The user experience is based on a three stage workflow. The user enters the nameplate generator on the start page. Here, the search bar is present to enter the aasx server address. Beyond that, formerly visited aasx server addresses will be provided as suggestions through the search bar. At the top of the page within the hamburger menu, a `Home` button is displayed, which always gives the user the possibility to return to the start page. The same effect can be reached by clicking on the nameplate generator logo in the header bar on the left side. A `About` button in the hamburger menu provides extra information around the application and the `GitHub` button leads to the GitHub repository of this project. After inserting the server address of the aasx server, the assets get loaded from it. Continuing from there, the user gets forwarded to the asset list. This list displays all assets that the server provides.

They can be identified via the asset id. If the chosen server supports the V3 api, the product images are displayed if they exist in the asset. This is not the case for V1 servers, since the older api does not support this feature. The user has the ability to search through the asset list by using the search bar at the top of the page. Furthermore, the user can now choose one asset to be displayed in detail in the asset view. In this screen, the user gets offered information of the asset. This includes all submodels and their contained data. The nameplate submodel is displayed separately on the left side of the screen while all other submodels can be inspected on the right hand side of the screen.

Furthermore, the possibility to download the nameplate for the asset in `.png` or `.svg` format is given. This can be utilized by clicking either the `Download SVG` or the `Download PNG` button. Right above these buttons is a display of a nameplate preview.

3.2 Requirements

DNG.GUI.001 Responsive and Compatible GUI

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 be compatible with multiple browsers.

DNG.GUI.002 Download buttons for SVG and PNG format

The nameplates are downloaded onto the user's device. A SVG or PNG version of the exact nameplate displayed on the page shall be downloaded. The format depends on which button is pressed.

DNG.GUI.003 Search functionality

The search functions allow the user to search for a certain asset or server. 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.

DNG.GUI.004 Navigation buttons

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.

DNG.GUI.007 Error Handling

The system has an error handling. When the server is down or does not exist, the user shall be notified.

DNG.USER.001 User-friendly

The application should be user-friendly, thus, a user with no experience with the website shall be able to use it effortlessly. 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.

DNG.LIC.001

The product is an open source software thus a license for publishing it is required.

3.3 Module Context

This module has two defined interfaces to display the asset list and the asset view. This enables the web interface to operate together with different modules, as long as the Information are provided in the following format.

The web interface presents the data to the user. On entering the server address and initializing the fetch by clicking the search icon, the data from the aasx server is requested by MOD02. The response from the server is refined and returned to MOD01. The data format that MOD01 accepts is the following. From this data, the web interface generates the asset list and the asset view.

```
[
  {
    // asset1
  },
  {
    // asset2
  },
  {
    // asset3
  },
  {
    // ...
  }
]
```

The asset mentioned in the format above are defined as follows. The following data is an example asset. On all levels of the json, keys can be added or removed depending on the level of detail of the asset. The web interface dynamically adapts to new keys being present or keys missing. Some urls have been shortened for presentation purposes.

```
{
  "idShort": "AAS_Type_CD55B20_50",
  "id": "www.example.com/ids/aas/7031_8082_3022_7912",
  "num": 8,
  "productImages": [
    "<url>"
  ],
  "Nameplate": {
    "idShort": "Nameplate",
    "id": "www.example.com/ids/sm/0000_4121_5022_2603",
    "ManufacturerName": "SMC",
    "ManufacturerProductDesignation": "Kompaktzylinder nach ISO 21287",
    "ManufacturerProductFamily": "C55",
    "OrderCode": "CD55B20-50",
    "SerialNumber": "",
    "YearOfConstruction": "2022",
    "Address": {
      "Department": "Kontakt zu SMC",
      "Street": "Boschring 13-15",
      "ZipCode": "63329",
      "City_Town": "Egelsbach",
      "State_County": "Deutschland",
      "VATNumber": "DE 0123456789",
      "AddressOfAdditionalLink": "info@smc.de",
      "Phone": {
        "TelephoneNumber": "+49 (0) 61 03 / 402 - 0",
        "TypeOfTelephone": ""
      }
    },
```

```
    "Markings": {
      "Marking00": {
        "MarkingName": "nach EU-Maschinen-Richtlinie",
        "FilePath": "<url>",
        "MarkingFile": "/aasx/Nameplate/CE_Marking_2016.png"
      },
      "Marking01": {
        "MarkingName": "nach EU-RoHS-Richtlinie",
        "FilePath": "<url>"
      },
      "Marking02": {
        "MarkingName": "nach EU-EMV-Richtlinie",
        "FilePath": "<url>",
        "MarkingFile": "/aasx/Nameplate/CE_Marking_2016.png"
      },
      "Marking03": {
        "MarkingName": "RCM Mark"
      },
      "Marking04": {
        "MarkingName": "c UL us - Listed (OL)"
      },
      "Marking05": {
        "MarkingName": "TÜV"
      }
    }
  },
  "ArticleInformation": {
    // more key-value pairs
  },
  "ContactInformation": {
    // more key-value pairs
  }
  // more submodels
}
```

4 Analysis

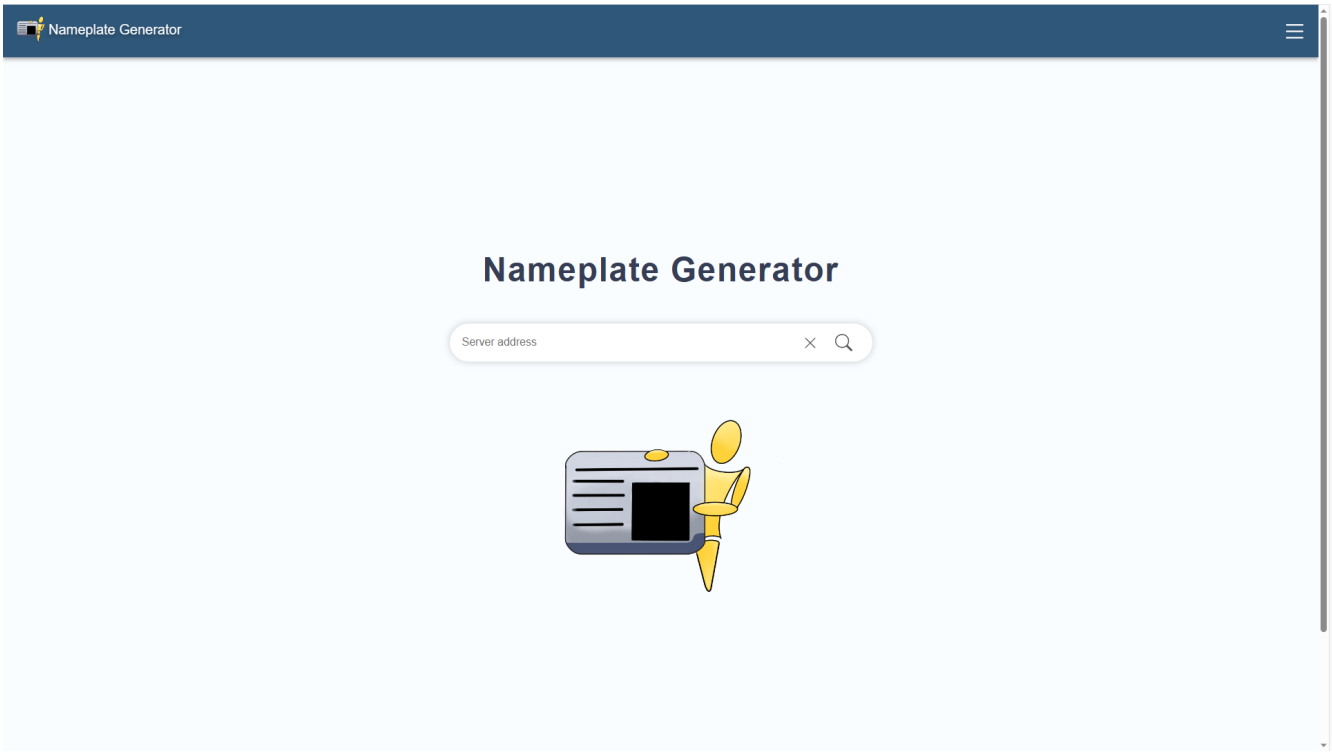
The focus of this module is to create an appealing and functional GUI which works with data from different modules. The entire process of fetching and refining data is moved to different modules. This increases the modularity of the application by keeping the web interface exchangeable, because it provides the visualization of the data separated from its processing. The web interface puts an emphasis on elegance in form of simplicity. The GUI is kept minimalistic to avoid confusion of the user. The inputs that user has to give are kept as low as possible. With these points, the user interface provides a positive user experience and supports the nameplate generator to create value for anyone using it.

5 Design

This three step workflow, start page - asset list - asset view, is depicted in the following images.

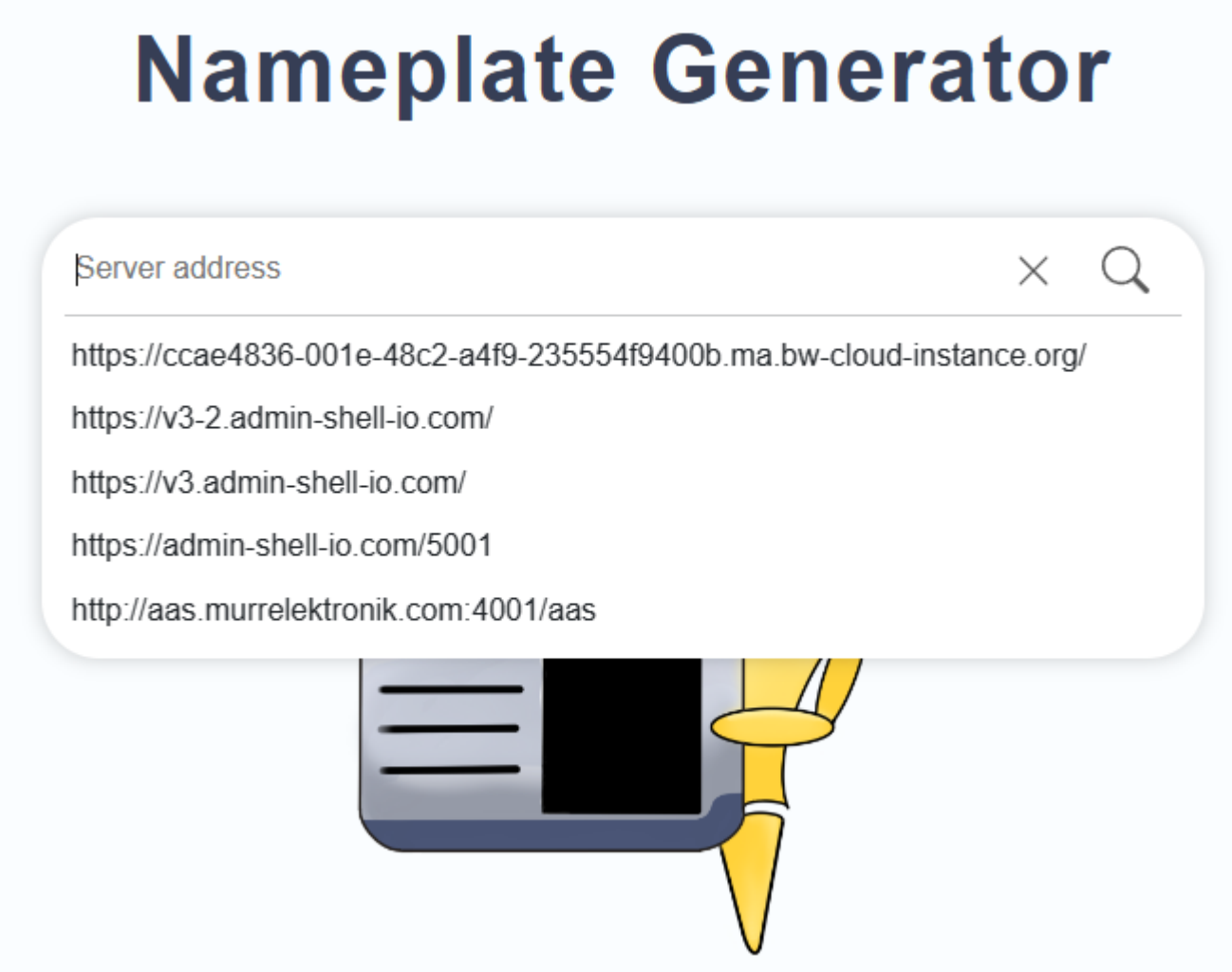
Start page:

The start page is the entrypoint of the workflow for the Nameplate Generator. Here, the user can insert the server address of the aasx server he wants to fetch data from.



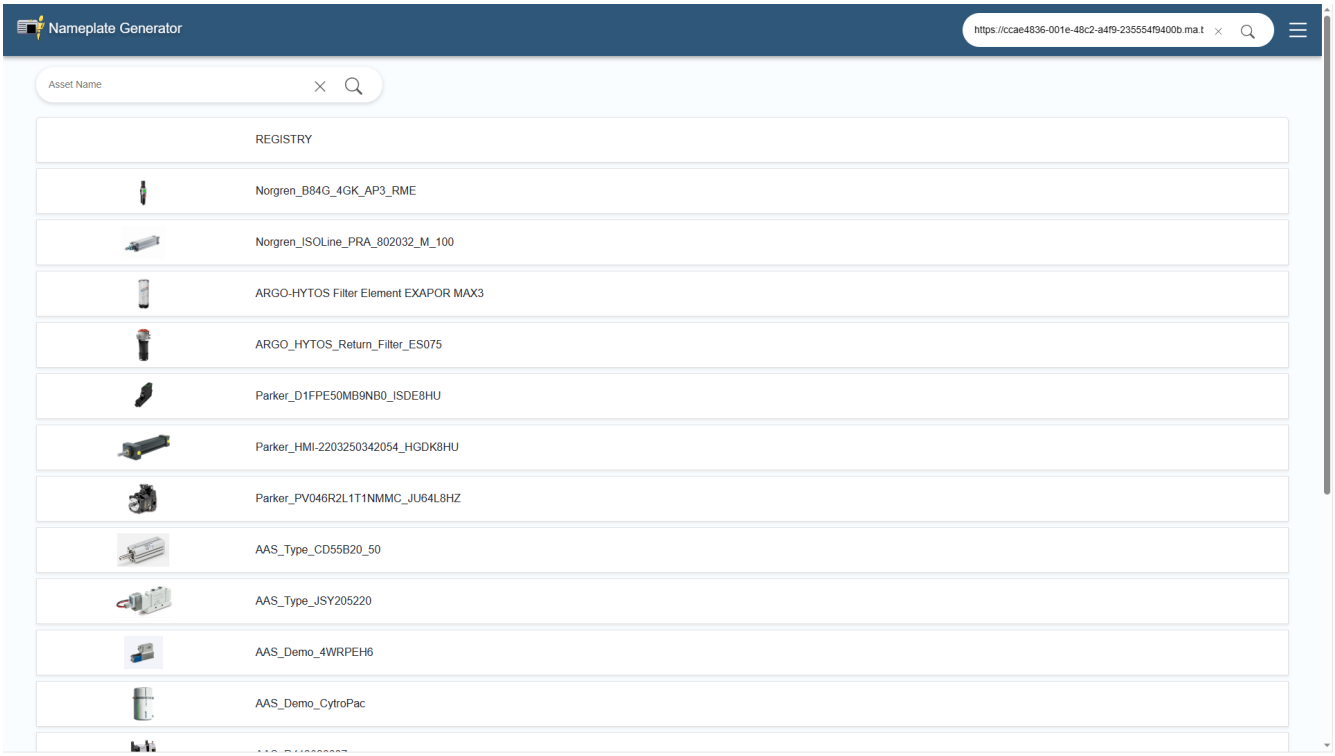
Search bar suggestions

In the suggestions of the search bar, recently used servers are suggested for a quicker entry into the workflow and an improved user experience.



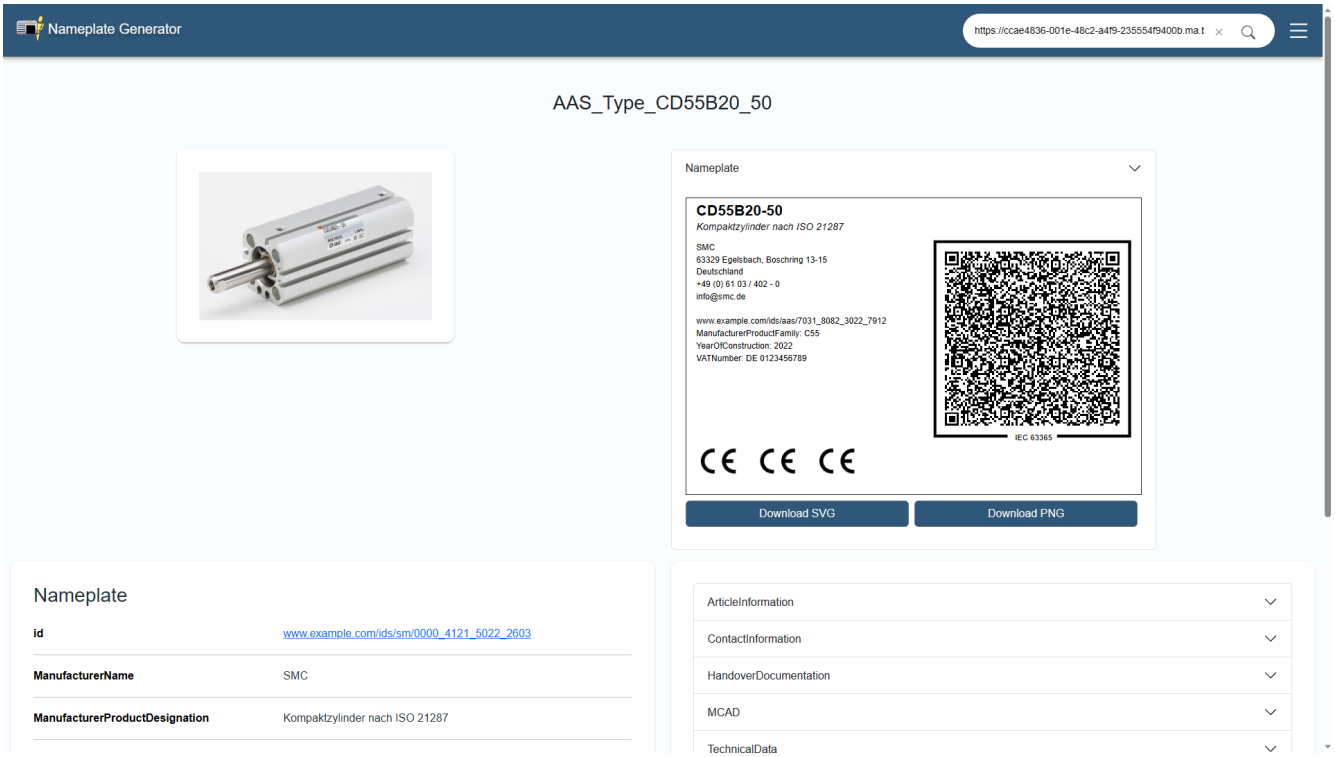
Asset list:

In the asset list, the user can pick a asset to inspect.



Asset view:

The asset view provides detailed information about the asset including all submodel information compressed into a set of accordions. Furthermore, a preview of the nameplate is shown together with the corresponding download buttons.



Example Nameplate:

This is an example of a nameplate generated by the Nameplate Generator. It provides all necessary information about the physical machine corresponding to the asset. It is conform to IEC 63365.

5.1 Risk

The front end is based on the open source library React. It has been created by Meta, formerly known as Facebook, and is published under the MIT license. This allows our team to use the library for our own purposes and to change it if needed. If Meta decided to change the license to a copyright, that would either cause license costs for the web interface or the GUI needs to be recode with a different library or framework, e.g. Angular. However, due to the great publicity of React and it having a serious open source competitor under the MIT license, Angular by Google, the risk of React changing its license to a copyright can be classified as low.

6 Implementation

MOD01 is written in plain React. It is separated internally into the following subcategories.

1. Home Screen
2. Asset List
3. Asset View
4. About Page
5. Navigation
6. Warnings

All subcategories are build in a responsive manner. This means, that the Nameplate Generator is build to work on any device from smartphone to desktop computer.

Home Screen

The home screen is implemented as React functional component, which can be found at `APPLICATION/src/Homescreen/HomeView.j` It is exported and is utilized in the `App.js` file, in which the application is composed using the different React components. The home screen makes use of another React functional component, the search bar. It is an important element of the homepage giving the user the possibility to retrieve data from a aasx server of his choice. This design was chosen to keep the search bar structurally separated and therefore keeping it exchangeable.

Asset List

The asset list consists of three React components, two being functional and one being a class component. The implementation of the asset list frontend can be found under `APPLICATION/src/AssetList`. The asset list is composed of two components, the `AssetList.js` and the `AssetListElement.js`. The first file implements the component representing the asset list as a whole. This component makes use of the asset list element. Again, this keeps the asset list element within the list as a structure modular and exchangeable.

The list view is responsible to detect updates in the frontend due to user interaction and update the web interface accordingly.

Asset View

The React components `AssetView.js` and `AssetData.js` together make up the asset view. The asset data constructs the accordions displayed in the asset view. These are integrated into the component `AssetView.js` which then can be used to display the data. The source code for this part of the web interface can be found at `APPLICATION/src/AboutPage`.

About Page

The about page is implemented by one React functional component located at `APPLICATION/src/AboutPage`. If the user is routed to the about page, it will be rendered displaying information about the Nameplate Generator.

Navigation

The source code for the navigation of the page can be found at `APPLICATION/src/Navigation`. The React component within `Navbar.js` creates the display of the navigation header. It is being utilized at the root of the application in `App.js` so that the navigation is always displayed on every page but is only implemented once. The concrete routing is defined in the `App.js` file. To enable the user to navigate back and forth throughout the workflow, the forwards and backwards buttons provided by the browser are supported in their natural functionality.

Warnings

The functional React component implemented in `APPLICATION/src/Warnings.js` is a universal warning display component. It finds use in e.g. the warning that is displayed when you retrieve data from a aasx server running the V1 api saying that images can not be loaded. Any warning text can be displayed at any given time through this functional component. In addition to that, it can also be used to display errors, e.g. if a server does not respond. The user gets feedback via a colored banner in the ui. This makes this component highly versatile and useful for future development.

7 Module Tests

In this section nearly all requirements will be tested separately on their functionality. Some requirements are not tested, because they have no function to test on.

7.1 Module Test Plan

Req. - ID	Functionality
DNG.GUI.001	Responsive and compatible GUI
DNG.GUI.002	Download menu for SVG and PNG format
DNG.GUI.003	Search functionality
DNG.GUI.004	Navigation buttons
DNG.GUI.007	Error handling

Requirement DNG.USER.001 User friendly is not listed since it is out of scope.

7.2 Module Test Report

Req. - ID	Pass / Fail	When failed: Observation	Tester
DNG.GUI.001	PASS		Erika Zhang
DNG.GUI.002	PASS		Maris Koch
DNG.GUI.003	PASS		Mika Kuge
DNG.GUI.004	PASS		Adrian Khairi
DNG.GUI.007	PASS		Mika Kuge

8 Summary

To sum up, the web interface is the module that provides the visual representation of the Nameplate Generator for the user. It focusses on providing pleasant yet efficient user experience by providing a simple three step workflow. This module focusses on keeping the visual representation separate from the data acquisition to keep the front end exchangeable and extendable. Hence, the web interface can work with any module providing the previously described data format.