Inhaltsverzeichnis

Abbildungsverzeichnis II
Vorwort III
Einleitung 1
1. Bestandsaufnahme und Zukunftsausblick 2
1.1. Die Vorversion 2
1.2. Die Grundidee einer Website 2
1.3. Rahmenbedingungen und Zielsetzung 2
2. Der Aufbau und die innere Struktur der Website 3
2.1. Vereinfachung von komplexen Themen 3
2.2. Die Grundlage 4
2.3. Wozu braucht es Funktionen? 4
2.3.1. Grundlegenden Funktionen 4
2.3.2. Die Funktion für das Hinzufügen von Video Parametern 5
2.3.3. Wie funktionieren dynamische Auswahlen? 6
2.3.4. Fehlerverarbeitung – der Error Handler 7
2.3.5. Bestätigungsfunktion - submitForm 7
2.3.6. Die Generation des Dokumentes – der PDF Print 8
3. Fazit 9
Quellenverzeichnis 10
Eigenständigkeitserklärung .....

Abbildungsverzeichnis

Abbildung 1 - Auszug aus dem originalen PDF-Dokument 2
Abbildung 2 – Skizze für die ersten Entwicklungsschritte 2
Abbildung 3 - Beispiel: Einbindung von Tooltip und Bildern 3
Abbildung 4 - Beispiel Bilder aus der Wahl des Videoanschlusses 3
Abbildung 5 – Code für die Darstellung des Videoanschlusses 3
Abbildung 6 – Vergleich der Website (links) und dem dazugehörigen Code aus ‚index.html' (rechts) 4
Abbildung 7 – Vergleich von Code und Darstellung für die Hardware Format Funktionalität 4
Abbildung 8 – Auszug aus dem alten PDF Dokument 5
Abbildung 9 – Auszug aus dem neuen PDF Dokument 5
Abbildung 10 – Vergleich von Code und Darstellung der UUT Video Input/Output Sektion 6
Abbildung 11 – Auszug des Codes für die dynamische Auswahl von Source/Sink 6
Abbildung 12 – Vergleich von Code und Darstellung der Fehlerverarbeitung 7
Abbildung 13 - Auszug aus dem Code der Bestätigungsfunktion 7
Abbildung 14 - Import PDF-Print Funktion 8
Abbildung 15 - Auszug aus der submitForm Funktion 8
Abbildung 16 – Beispiel für Koordinaten innerhalb der PDF-Print Funktion 8

Vorwort

Diese Arbeit entstand im Rahmen der ersten Praxisphase bei Göpel Electronics. Das Thema "Entwicklung einer interaktiven Website zur Erfassung und Vereinfachung technischer Parameter für Automotive-Video-Projekte, unter Beachtung nötiger Abhängigkeiten." war nicht unbedingt als Grundlage für die erste Projektarbeit gedacht, entwickelte sich jedoch im Rahmen der Aufgabe und meiner Freude an der Umsetzung dazu.

Die Website sollte in naher Zukunft auch ihren Platz auf der Göpel Startseite finden, wahrscheinlich aber nicht bis zum Abgabetermin dieser Arbeit.

Im Laufe der Entwicklung beschäftigte ich mich ausgiebig mit Webentwicklung, Plugins in VS Code und den Programmiersprachen: HTML, JavaScript und CSS. Alles war komplett neu für mich, daher bin ich umso stolzer auf das Ergebnis.

In dieser Projektarbeit schildere ich die Herausforderungen, die ich bewältigen musste, und wie ich mich Schritt für Schritt mit neuen Funktionen und Konzepten vertraut gemacht habe.

Der Code und das vorrausgehende sowie die erstellten PDF-Dokumente sind über den folgenden Link herunterladbar, sollte das Testen der Website auf dem lokalen Rechner verlangt sein.

www.github.com/jakob-work/Projektarbeit1

Einleitung

Im Umgang mit Kund*innen, ist Zeit ein kritischer Faktor - nicht, weil sie fehlt, sondern weil sie meist durch fehlerhafte oder aufwändige Prozesse verloren geht. Das Unterbreiten eines Angebotes sollte eine simple Aufgabe sein, welche innerhalb weniger Minuten abgeschlossen werden kann. Wir haben uns bisher darauf verlassen, dass die Kund*innen selbst genau wissen, was sie benötigen, anstatt unsere Fachkenntnis zu nutzen und ihnen so ein passendes Angebot zu machen.

Ziel war es, eine benutzerfreundliche Website zu entwickeln, die die bisherige Methode in Form eines PDF-Dokumentes ablöst. Die Struktur und Inhalte der Website orientieren sich an diesem Dokument, sodass der allgemeine Aufbau ähnlich bleibt, die Oberfläche wurde jedoch durch Darstellungen und dynamische Auswahlen erweitert. Der Fokus dabei lag auf einer Vereinfachung des gesamten Themenkomplexes für Kund*innen ohne fachliche Expertise, sowie einer effizienteren Einbindung in unseren Arbeitsablauf. Hintergrund dafür sind häufige Rückfragen bezüglich dieses Dokumentes, welche in Zukunft durch die Website hoffentlich minimiert werden.

Der Lösungsansatz bestand darin, die Nutzer*innen Schritt für Schritt durch alle relevanten technischen Parameter zu führen. Mithilfe dynamischer Auswahlfelder werden nur die jeweils passenden Optionen eingeblendet, um den Prozess zu vereinfachen und Eingabefehler weitgehend zu vermeiden. Um das bestehende PDF-Dokument dennoch beizubehalten, wird dieses am Ende des Prozesses in abgeänderter Form mit den von den Nutzer*innen ausgewählten Parametern automatisch ausgefüllt und zum Download bereitgestellt.

1. Bestandsaufnahme und Zukunftsausblick

1.1. Die Vorversion

Für eine lange Zeit wurde die Erfassung von Parametern für Automotive-Video-Projekte über eine PDF (Abbildung 1) abgewickelt. Dieses Formular war ursprünglich als einfache Lösung gedacht, wurde aber im Laufe der Zeit nicht überarbeitet und entspricht somit nicht mehr den heutigen Anforderungen. Wie in Abbildung 1 zu sehen, erfolgte jede Informationseingabe[1], bis auf wenige Ausnahmen[2], manuell und ohne Vorgaben.

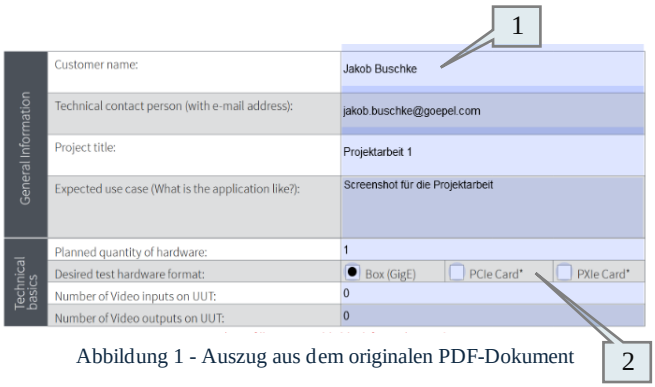

Abbildung 1 - Auszug aus dem originalen PDF-Dokument

1.2. Die Grundidee einer Website

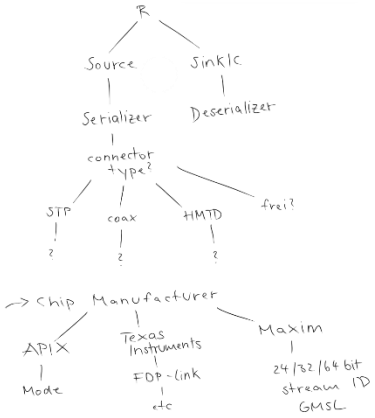

Abbildung 2 – Skizze für die ersten Entwicklungsschritte

Die Ausgangslage war folgende: eine PDF „ATS_Parameter-Checklist_Video_Projects“; ein einfaches Baumdiagramm (**Fehler! Verweisquelle konnte nicht gefunden werden.**) und die Informationen aus unserem Unternehmens-Wiki.

Das Projekt entstand im Rahmen einer Weiterentwicklung von internen Prozessen aber in aller erster Linie als gutes Projekt für die Praxisphase. Die Grundidee war, dass der Aufbau der Website ungefähr die Struktur der PDF widerspiegelt, aber durch: Dropdown-Menüs; Knöpfe; Bilder und dynamische Ein- und Ausblendungen erweitert wird. Das Einbinden von weiteren Funktionen wie einer Fehlerbearbeitung, Druck einer PDF als Resultat und dynamischen Auswahlen war das Hauptziel. Vorher bot das PDF-Dokument im Grunde keine Hilfestellungen, dieses Problem und viele weitere löste das neue Konzept weitestgehend.

1.3. Rahmenbedingungen und Zielsetzung

Vor Beginn der eigentlichen Umsetzung entwickelte ich eine Art Konzept für die Entwicklung beziehungsweise setzte Ziele, die erreicht werden sollten. Der Code sollte übersichtlich und einfach zu bearbeiten, sowie für die Zukunft auch erweiterbar sein. Die Benutzeroberfläche auf der Website sollte gut strukturiert und nutzerfreundlich sein, außerdem sollte sie genügend Informationen für fachfremde Personen bieten damit diese mit geringem Aufwand ein vollständig ausgefülltes Dokument erhalten. Aus diesem Grund plante ich von Anfang an mit Tooltips, bildlichen Darstellungen und dynamischen Auswahlfeldern.

Während der Entwicklung versuchte ich auf zukunftssichere Methoden zurückzugreifen, bewusste wurde auf externe Lösungen verzichten und mit einfachen Mitteln gearbeitet wie HTML, CSS und JavaScript. Auch die wenigen verwendeten Bibliotheken wurden auf ihre Langlebigkeit geprüft, damit die daraus benutzten Funktionalitäten so lange wie möglich in Takt bleiben.

Diese Entscheidungen ermöglichen eine flexible, erweiterbare und hoffentlich zukunftssichere Grundstruktur für diese komplexe Aufgabe.

Es sind bereits zukünftige Änderungen geplant, wie beispielsweise eine direkte Versendung des PDF-Dokumentes an den Göpel Vertrieb oder Support. Warum eine solche Funktionalität noch nicht möglich war, wird in einem der folgenden Kapitel erläutert.

Welche Funktionen bereits entwickelt wurden und wie ihre Einbindung verlaufen ist, beschreibt Kapitel 2 umfangreich.

2. Der Aufbau und die innere Struktur der Website

2.1. Vereinfachung von komplexen Themen

Dieses Kapitel beschäftigt sich weitestgehend mit den Vereinfachungen welche tatsächlich auf der Website zu erkennen sind. Wie der Name schon sagt, besprechen wir Bilder, Tooltips und weitere Erweiterungen der Website, welche den Nutzer*innen einen möglichst großen Nutzen bieten sollen. Dabei spielt nicht nur Zeitersparnis eine Rolle, sondern auch das Entwickeln eines Verständnisses für dieses Themengebiet.

In Abbildung 3 zu sehen, ist die Auswahl des ersten Parameters zu sehen. Die Auswahl des *desired hardware formats*, also des gewünschten Formats der Hardware, bildet die Grundlage für die restlichen Parameter und vor allem für ihre Einbindung. Um den Nutzer*innen eine möglichst passende Grundlagen zu schaffen, helfen Tooltip[1] und eine bildliche Darstellung[2]. Ein Tooltip ist eine Beschreibung eines Elementes, welche meist als Popup aufgerufen wird. Das Aufrufen dieses Tooltips funktioniert, indem man mit der Maus über das blaue (i)[3] fährt. Er enthält eine Beschreibung über die Funktion und den Einsatz der einzelnen Formate. Fachfremden Nutzer*innen soll somit gewährleistet werden, dass sie die richtige Entscheidung treffen.

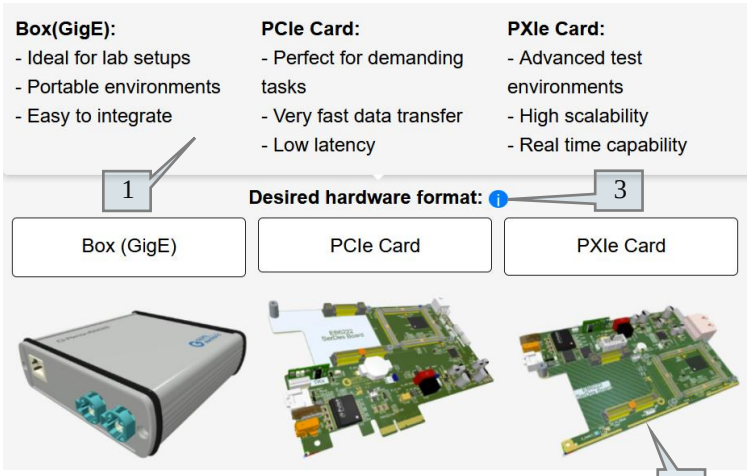

Abbildung 3 - Beispiel: Einbindung von Tooltip und Bildern

Abbildung 4 - Beispiel Bilder aus der Wahl des Videoanschlusses
links nach rechts: STP, COAX, HMTD

Sollte beispielsweise eine fachfremde Person physisch ein Kabel vorliegen haben, kann sie mit Hilfe der Bilder aus Abbildung 4 die richtige Wahl für den benötigten Videoanschluss treffen.

Die Nutzbarkeit wird durch Bilder und Tooltips weitgehend verbessert, das Auswählen von Parametern wird außerdem vereinfacht und hat gegenüber ausschließlich textbasierten Beschreibungen kaum Nachteile. Die Umsetzung der Tooltips und Bildintegration innerhalb des Codes wurde so angefertigt, dass zukünftige Erweiterungen, wie neue Steckertypen, oder Änderungen problemlos durchgeführt werden können, ohne etwas an der bestehenden Struktur zu verändern.

Beispielsweise weitere Steckertypen hinzuzufügen wäre schnell erledigt. In Abbildung 5 sieht man den Aufbau des html-Codes für die in Abbildung 4 dargestellten Bilder. Innerhalb der Ordnerstruktur des Projektes gibt es den Ordner: *images*, welcher alle auf der Website zu sehenden Bilder enthält. Ein Bild des neuen Steckertyps kann dorthin verschoben werden und dann über diesen Code[1] hinzugefügt werden. Danach noch ein weiterer Knopf der den Namen[2] enthält und so wurde in weniger Schritten ein neuer Knopf mit entsprechendem Bild erstellt. Diese einfache Struktur ermöglicht das schnelle Bearbeiten und Erweitern der Funktionalität und bietet einen großen Mehrwert für die Nutzer*innen.

```
<div class="form-group video-connector-type" style="display: none;">
<label>Video connector type:</label>
<div class="buttons">
    <button type="button" class="btn option" data-value="STP">STP</button>
    <button type="button" class="btn option" data-value="Coax">Coax</button>
    <button type="button" class="btn option" data-value="HMTD">HMTD</button>
    <button type="button" class="btn option" data-value="Other">Other</button>
</div>
<div class="video-connector-images" style="margin-top: 10px;">
    <img src="images/STP.png" alt="STP" class="video-connector-image">
    <img src="images/Coax.png" alt="Coax" class="video-connector-image">
    <img src="images/Hmtd.png" alt="HMTD" class="video-connector-image">
```

2

1

Abbildung 5 – Code für die Darstellung des Videoanschlusses

2.2. Die Grundlage

Die Grundstruktur der Website bilden Anfangs drei Dateien, später dann vier. Üblicherweise handelt es sich hierbei um eine ‚index.html' – Datei geschrieben in HTML, enthält Text und Bilder. Die Datei ‚style.css' geschrieben in CSS, welche das Layout der Seite definiert, schließt Textgröße, Farben und der Gestaltung von Bedienelementen wie Knöpfen oder Dropdown Menüs ein. Abschließend die Datei ‚script.js' geschrieben in JavaScript, welche alle Funktionalitäten implementiert, beispielsweise: Knöpfe, Dropdown Menüs sowie das Ein- und Ausblenden von Texten und Bildern. Am Ende der Entwicklung überarbeitete ich die Code Struktur, überarbeitete und ordnete den Großteil noch einmal damit die zukünftige Bearbeitung einfacher ausführbar ist. Die Lesbarkeit der ‚script.js' Datei war mühsam, vor allem durch lange komplizierte Funktionen wie die PDF-Print Funktion. Aus diesem Grund erstellte ich für diese Funktion eine separate Datei namens ‚pdfPrint.js' die dann innerhalb ‚script.js' gerufen werden kann und die Funktionalität bei verbesserter Übersicht gleichbleibt.

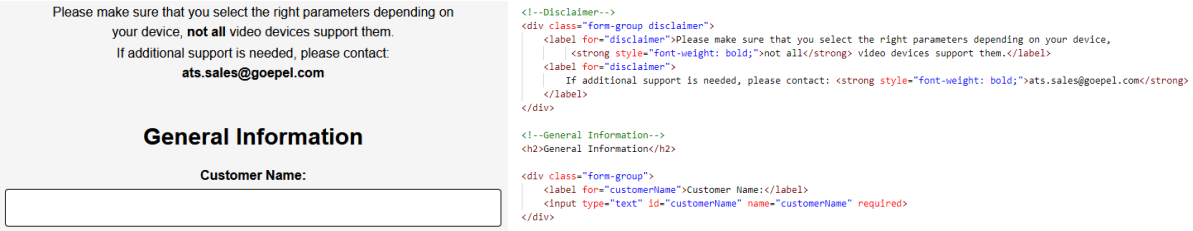

Abbildung 6 – Vergleich der Website (links) und dem dazugehörigen Code aus ‚index.html' (rechts)

2.3. Wozu braucht es Funktionen?

Eine interaktive Benutzeroberfläche war der wichtigste Aspekt in der Entwicklung der Website, um das umzusetzen werden viele unterschiedliche Funktionen benötigt. In diesem Teil der Projektarbeit werden die wichtigsten Funktionen erläutert und die Herangehensweise für die Entwicklung beschrieben.

2.3.1. Grundlegenden Funktionen

Den Grundstein legen die Knöpfe, Schieberegler und Dropdown-Menüs, daher gehen wir kurz auf ihre Funktionalität ein.

Um das zu vereinfachen, werde ich den Aufbau eines Knopfes anhand Abbildung 7 erläutern. Von oben nach unten zu sehen sind: der Aufbau der Schaltfläche innerhalb des html-Script, die Darstellung auf der Website und die Funktion innerhalb der script.js-Datei. Dort sieht man die Einbindung von Texten[1], Bildern[2] und Attributen wie *class*, *type* oder *name*. Attribute bieten uns die Möglichkeit, das Aussehen oder Verhalten des Bedienelementes zu definieren. Eine einfache Lösung die Funktionalität festzulegen, bietet *type*. In unserem Fall benutzen wir den Typen *radio*, jener ermöglicht, dass jeweils nur eine Option pro Auswahl gewählt werden kann. Mithilfe von *class*, lassen sich mehrere Bedienelemente in Gruppen zusammenfassen, um ihnen in der style.css-Datei ein einheitliches Aussehen zuzuweisen. Damit mehrere Knöpfe vom gleichen Typen in Abhängigkeit voneinander funktionieren, vergeben wir ihnen einen *name*, dies ermöglicht das jeweils nur ein Button pro Gruppe ausgewählt werden kann.

Abbildung 7 – Vergleich von Code und Darstellung für die Hardware Format Funktionalität

2.3.2. Die Funktion für das Hinzufügen von Video Parametern

Die Abkürzung UUT steht für *Unit Under Testing*, bezeichnet hierbei also das Gerät an jenem Tests und Messungen ausgeführt werden sollen.

Dieser Teil der Website beinhaltet die wichtigsten Parameter, daher lag das Augenmerk vor allem auf einer Komprimierung und Vereinfachung des gesamten Themenkomplexes. Die alte Erfassungsmethode umfasste jeden einzelnen Parameter, auch solche zwischen denen man im Verlaufe des Ausfüllens entscheiden muss. Das führte zu einem sehr unübersichtlichen PDF-Dokument, da sehr viele Antwortbereiche unausgefüllt oder falsch ausgefüllt wurden. Im folgenden Abschnitt werde ich kurz auf die Unterschiede zwischen der alten und der neuen Version eingehen, sowie meine Lösungsansätze für die genannte Probleme liefern.

Meine Idee bestand darin, für jeden unterschiedlichen Fall ein separates PDF-Dokument zu erstellen. Die Unterschiede zwischen der alten Version (Abbildung 8) und der neuen Version (Abbildung 9) sind sofort erkennbar, wie man erkennen kann enthält das alte Dokument jede Information zugehörig zu *Source*, auch wenn die Nutzer*innen der *Sink* ausgewählt haben sollten.

Source und *Sink* stehen hierbei für die Signalrichtung, übersetzt bedeuten diese Quelle und Senke. Source wird immer dann benutzt, wenn ein Signal versendet werden muss. Verwendung findet das beispielsweise beim Testen eines digitalen Tachometers. Sink wiederum, wird verwendet, wenn ein Signal empfangen und verarbeitet werden soll. Exemplarisch beim Testen von Rückfahrkameras, diese werden darauf getestet, ob sie ein Videosignal sendet und welches Format und welche Qualität dieses hat.

Weiterhin in Abbildung 8 zu sehen sind alle Parameter der verschiedenen Mikrochip-Hersteller, obwohl man pro Einheit nur einen auswählen kann. Das neue Dokument löst diese Probleme und erreicht somit eine optimale Übersicht. Abbildung 9 stellt einen Fall dar, in welchem *Sink* als Signalrichtung sowie *Apix* als Mikrochip-Hersteller gewählt wurden. Für jede Kombination gibt es ein separates PDF-Dokument, Vorarbeit für die PDF-Druck Funktion, welche in einem der folgenden Abschnitte vertieft beschrieben wird.

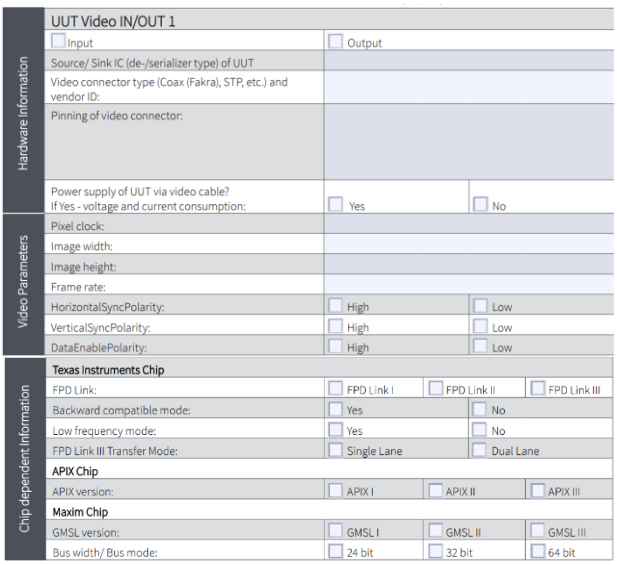

Abbildung 8 – Auszug aus dem alten PDF Dokument

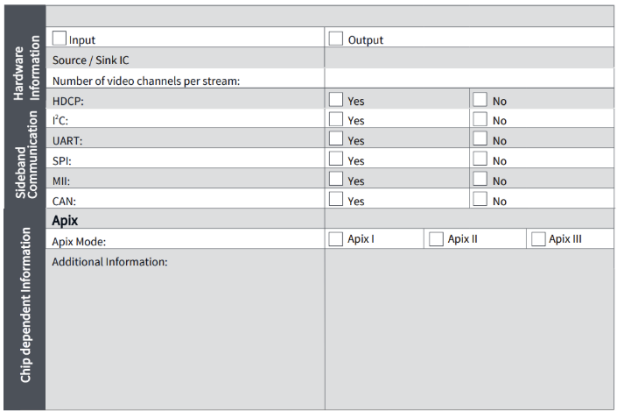

Abbildung 9 – Auszug aus dem neuen PDF Dokument

Das Bearbeiten und Erstellen jeder Datei beanspruchte viel Zeit, machte jedoch das Weiterarbeiten und Bearbeiten in der Zukunft deutlich angenehmer. Damit können einzelne Anpassungen schnell gemacht werden, ohne die gesamte Struktur zu verändern. Natürlich birgt sie aber auch Nachteile, sollte sich etwas innerhalb von zu Beispiel *Sideband Communication* verändern, muss jede Datei einzeln verändert werden.

Als ersten Lösungsansatz verfolgte ich das Nachbilden des alten Dokumentes, dieser Ansatz entwickelte sich für meine JavaScript Kenntnisse doch zu einem zu großen Hindernis. Entwicklungsarbeit ist nie abgeschlossen, damit bleibt das erstmal ein Projekt für die Zukunft.

Ein neuer „UUT Video IN/OUT“ wird mit Hilfe eines Knopfes unterhalb der grundlegenden technischen Informationen hinzugefügt. Falls ein zweiter hinzugefügt werden sollte erfolgt die Nummerierung auf der Website und innerhalb des gedruckten Dokumentes dynamisch. Der Aufbau der Funktion ist einfach gehalten, sie beinhaltet

alle Elemente als html-Struktur, das macht die Bearbeitung nutzerfreundlich. Jeder weitere UUT Video Abschnitt wird unterhalb des vorherigen Elementes eingefügt und umfasst alle Dropdown Menüs, Knöpfe und Parameter, die für die Auswahl notwendig sind.

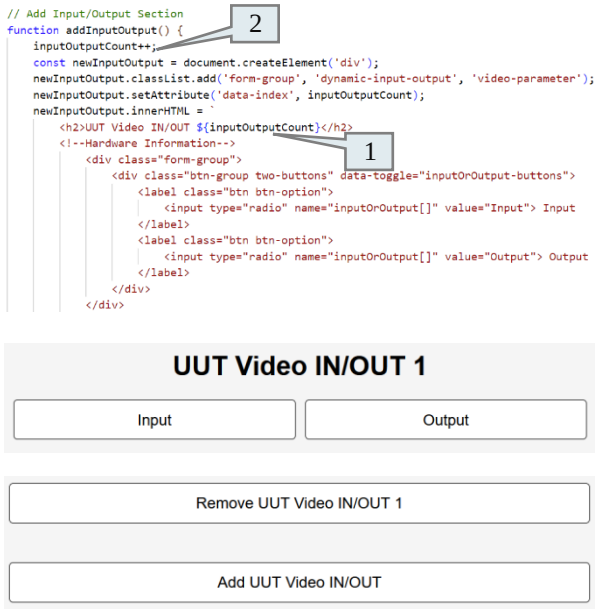

Abbildung 10 – Vergleich von Code und Darstellung der UUT Video Input/Output Sektion

In Abbildung 10 (oben) ist ein Ausschnitt aus script.js zu sehen, dieser stellt den Anfang der Funktion dar. Der untere Teil zeigt die Darstellung auf der Website, hier zu erkennen sind einmal die Überschrift und der erste Parameter, sowie die Knöpfe zum Löschen der ersten Sektion und zum Hinzufügen einer zweiten Sektion.

Wie schon erwähnt erfolgt die Nummerierung auf der Website und innerhalb der PDF dynamisch, die Funktionalität dafür hängt mit dem *inputOutputCount*[1] zusammen. Dieser wird bei jeder Betätigung des Add-Knopfes um eins erhöht[2], bei jeder Löschung einer Sektion um eins niedriger.

Der Rest der Funktion besteht zum Großteil nur noch aus der html-Struktur, die Einbindung der Funktionalität der Knöpfe war überraschend einfach. Die bereits definierten Funktionalitäten für die Knöpfe und Dropdown-Menüs konnten plug-and-play artig benutzt werden.

2.3.3. Wie funktionieren dynamische Auswahlen?

Die Funktionen für die dynamischen Auswahlen und das damit verbundene Anzeigen von bestimmten Textgruppen konnte ich simpel umsetzen. Für die Source/Sink sowie die Mikrochip Hersteller Auswahl verwendete ich eine einfache *if-else* Schleife für Source/Sink und *if-else if* Schleifen für die Mikrochip Hersteller. Je nach Auswahl prüfen diese Schleifen welche Möglichkeit ausgewählt wurde und leitet diese dementsprechend an die Print-Funktion weiter.

Sollte Source ausgewählt worden sein, werden unterhalb des dazugehörigen Dropdown-Menüs weitere Parameter angezeigt, diesen Vorgang des Einblendens löste ich ebenfalls mit Hilfe von einer if-else Schleife.

In Abbildung 11 ist ein Auszug von der *if-else if* Schleife der Funktion für die Source und Sink-Auswahl zu sehen. Dort kann man schnell den einfachen Aufbau erkennen. Sollte man auf der Website *Source* auswählen, wird automatisch *Serializer* ausgewählt. Außerdem werden die zugehörigen Parameter eingeblendet. Sollte die Auswahl jedoch auf *Sink* fallen, wird *Deserializer* ausgewählt und die Parameter zugehörig zu *Source* werden ausgeblendet. Die Schlüsselwörter *block*[2] und *none*[1] haben Einfluss auf die *display* - Variable. Diese bestimmt die Darstellung auf der Website, beispielsweise für den Wert *none* wird nichts angezeigt. Für den Wert *block* ist vordefiniert, dass er die gesamte Breite einnimmt und in einer neuen Zeile beginnt. Man kann jedoch innerhalb des style.css die Darstellung dieses Elementes verändern. Mit Hilfe dieser einfachen Funktionalität lässt sich ein doch komplexes Thema leicht lösen, weiterhin ist diese zukunftssicher formatiert sodass in Zukunft weitere Parameter problemlos hinzugefügt oder entfernt werden können. Auch eine weitere Auswahl kann über eine zweite *else if* Schleife eingefügt werden, sollte dies in Zukunft verlangt werden.

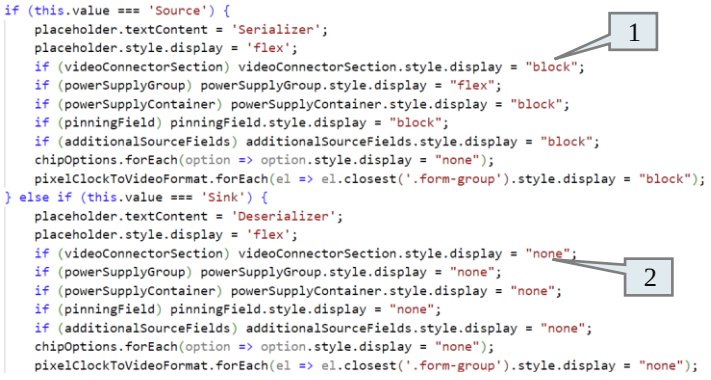

Abbildung 11 – Auszug des Codes für die dynamische Auswahl von Source/Sink

2.3.4. Fehlerverarbeitung – der Error Handler

Die Fehlerverarbeitung verfolgt einen simplen Ansatz, es sollte nichts unausgefüllt bleiben. Um diese Funktionalität zu erreichen, überprüft sie ob alle Felder einen Wert haben, sollte eines keinen Wert besitzen wird jenes Feld auf dem Bildschirm der Nutzer*innen zentriert und rot markiert (Abbildung 12, unten). Sollte dies der Fall sein, wird natürlich keine PDF erzeugt. Falls gewisse Felder ausgeblendet werden, sollten diese nicht überprüft werden. Hierfür nutze ich *offsetParent[1]*, eine Eigenschaft, welche dafür sorgt, dass nur sichtbare Felder von der Fehlerverarbeitung berücksichtig werden. Durch die große Menge an dynamischen Feldern und Dropdown-Menüs, die vorliegen, war dieser Zusatz nicht zu ignorieren. Zusammengefasst lässt sich sagen, dass der errorHandler eine der wichtigsten Funktionen der ganzen Website ist, er sorgt für ein vollständiges Ausfüllen der nötigen Parametern und macht es den Nutzer*innen einfach, unausgefüllte stellen sofort zu erkennen. Unsere Mitarbeiter*innen sind darauf angewiesen ein lückenloses Dokument zu erhalten, damit die Bearbeitung ohne Rückfragen umgehend erledigt werden kann. Die meisten Ideen für die Umsetzung dieses Projektes folgten diesem Ziel, damit Zeit und Arbeit gespart werden kann.

```
// Highlight Errors
function highlightErrors() {
    const form = document.getElementById('videoProjectForm');
    const inputs = form.querySelectorAll('input:required, select:required, textarea:required');
    const visibleInputs = Array.from(inputs).filter(input => input.offsetParent !== null);
    let firstErrorElement = null;

    visibleInputs.forEach(input => {
        if (!input.value.trim()) {
            input.classList.add('error-highlight');
            if (!firstErrorElement) {
                firstErrorElement = input;
            }
        }
    });
```

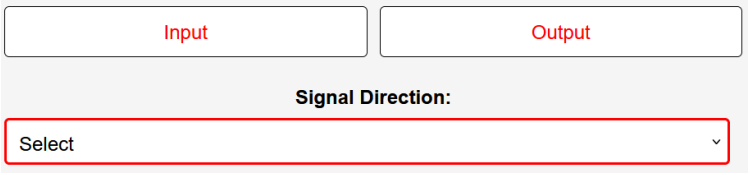

Abbildung 12 – Vergleich von Code und Darstellung der Fehlerverarbeitung

2.3.5. Bestätigungsfunktion - submitForm

Wenn letztendlich alles vollständig ausgefüllt wurde, ruft man die submitForm Funktion, indem man auf der Website den Submit Knopf drückt. Nachdem dieser betätigt wurde, müssen die Nutzer*innen noch bestätigen, dass sie wirklich den submit Knopf drücken wollten und es nicht nur ein Versehen war. Die zusätzliche Bestätigung gibt den Nutzer*innen noch einmal die Möglichkeit ihre Angaben zu überprüfen, bevor diese dann final verarbeitet werden. Bei der Erstellung der Funktion nutzte ich bewusst die *async*[1] Funktion, sie bietet die technische Grundlage für eine serverseitige Einbindung. In Zukunft soll es beispielsweise die Alternative geben, die Datei direkt, nachdem sie ausgefüllt und erstellt wurde, an unseren Support beziehungsweise Vetrieb weiterzuleiten. Zurzeit verläuft der Sendungsvorgang noch über den Download der PDF und Email. Eine Funktionalität, die eine automatische Versendung möglich macht. Die Entwicklung dafür stellte doch eine große Herausforderung dar, für den Sendungsprozess wäre eine *relay* Email, also eine eigene Email Adresse nur für das Weiterleiten des Dokumentes an unseren Support oder Vertrieb nötig gewesen. Außerdem müsste sichergestellt werden, dass die Daten der Nutzer*innen nicht unverschlüsselt übertragen werden. Die Entwicklung dafür war in dem kurzen Zeitraum ohne Vorkenntnisse einfach nicht möglich. Abschließend lässt sich sagen, dass diese Funktion eine der wichtigsten innerhalb des script.js ist. Sie verbindet die Fehlerbearbeitung direkt mit dem Druck des Dokumentes und stellt sicher, dass alle Parameter vollständig vorliegen. Der Aufbau der Funktion, also die Teilung in einzelne Funktionen bietet gute Möglichkeiten genannte Erweiterungen in Zukunft problemlos hinzuzufügen.

```
// Submit Form
document.getElementById('submitForm').addEventListener('click', async function (e) {
    e.preventDefault();

    const inputs = form.querySelectorAll('input:required, select:required, textarea:required');
    const visibleInputs = Array.from(inputs).filter(input => input.offsetParent !== null);
    let hasErrors = false;

    visibleInputs.forEach(input => {
        if (!input.value.trim()) {
            input.classList.add('error-highlight');
            hasErrors = true;
        }
    });
    if (hasErrors) {
        highlightErrors();
        return;
    }
    const userConfirmed = confirm("Submitting form, are you sure?");
    if (!userConfirmed) {
        return;
    }
    console.log("Submit button clicked");
```

Abbildung 13 - Auszug aus dem Code der Bestätigungsfunktion

2.3.6. Die Generation des Dokumentes – der PDF Print

Nachdem wir den Großteil der wichtigen Funktion nun beleuchtet haben, kommen wir zu der umfangreichsten Funktion. Die PDF-Print Funktion kam zuerst innerhalb des script.js vor, um jedoch eine gute Übersichtlichkeit zu bewahren, erstellte ich eine neue Datei allein nur für sie. Diese kann dann sehr simple innerhalb des script.js gerufen werden, die Nutzung verläuft exakt wie vorher.

Abbildung 14 zeigt die ersten zwei Zeilen des script.js, zu sehen ist der Import der Funktion *generatePDF* aus der pdfPrint.js Datei. Im zweiten Schritt wird *generatePDF* global verfügbar gemacht, mit Hilfe von *window*[1] kann man diese dann auch außerhalb des script.js aufrufen. In diesem Fall nicht notwendig, könnte allerdings in Zukunft benötigt werden.

```
import { generatePDF } from './pdfPrint.js';
window.generatePDF = generatePDF;
```

1

Abbildung 14 - Import PDF-Print Funktion

Nachdem wir *generatePDF* importiert haben, kann die PDF für den Druck vorbereitet werden. Damit die Datei überhaupt generiert werden kann, müssen zuerst alle Parameter vollständig vorhanden sein. Verbunden mit der *submitForm* Funktion und dem *errorHandler*, werden die Auswahlen kontrolliert und sollte etwas nicht vorhanden sein, wird die Erstellung der PDF abgebrochen. Innerhalb von *formData*[2] befinden sich alle notwendigen Parameter, diese werden der Reihe nach überprüft, sollte einer unausgefüllt sein wird dies für den Entwickler innerhalb der Konsole[3], jedoch auch für die Nutzer*innen auf der Website dargestellt.

2

```
const formData = {
    chipManufacturer,
    sinkOrSource,
    customerName,
    contactPerson,
    projectTitle,
    useCase,
    plannedQuantity
};

3

console.log('Form Data:', formData);

const additionalPdfPaths = [];
await generatePDF(formData, additionalPdfPaths);
```

Abbildung 15 - Auszug aus der submitForm Funktion

6

Desired test hardware format: | X Box (GigE) | PCIe Card* | PXIe Card*

```
const testHardwareFormat = document.querySelector
('input[name="testHardwareFormat[]"]:checked')?.value || '';
const hardwareFormatPositions = {
    'Box (GigE)': { x: 328, y: 477 },
    'PCIe Card': { x: 412, y: 477 },
    'PXIe Card': { x: 498, y: 477 },
};
if (testHardwareFormat in hardwareFormatPositions) {
    firstPage.drawText('X', {
        ...hardwareFormatPositions[testHardwareFormat],
        size: fontSize,
        font: sourceSansProFont,
        color: fontColor,
    });
}
```

5

1

2

3

4

Abbildung 16 – Beispiel für Koordinaten innerhalb der PDF-Print Funktion

Das Drucken des Textes auf die PDF stellte sich als einfache, aber auch zeitintensive Aufgabe heraus. In Abbildung 16 kann man den Aufbau sehen, die Positionierung des Textes erfolgt über Koordinaten[1]. Die Textgröße[2], die Schriftart[3] sowie die Farbe[4] des Textes deklarierte ich vorher, sodass sie sich für alle vorhandenen Parameter einheitlich ändern lassen, diese Methode wiederholte sich für jeden Parameter.

Die Funktion überprüft welches *testHardwareFormat* ausgewählt wurde[5] und druckt dann ein Kreuz[6] an der entsprechenden Position.

Für die Umsetzung des Druckvorgangs wählte ich die Bibliothek *pdf-lib*. Diese bietet mit *drawText*[7] die besten Möglichkeiten, Text zu positionieren. Html und CSS benutzen Layout basierte Positionierung, diese ist abhängig von beispielsweise: Margin, Padding und automatischen Zeilenumbrüche – alle dieser Eigenschaften hängen mit den vorherigen Eingaben zusammen. Damit verändert sich also der gesamte Aufbau, wenn nur eine einzige Änderung getätigt wird. Die Funktionalität von drawText basiert nicht auf den vorherigen Eingaben, jede Eingabe besitzt eigene Koordinaten und kann frei platziert werden. Layout basierte Platzierung kann im Falle einer Veränderung des PDF-Dokumentes, auf welches der Text gedruckt werden soll, schneller angepasst werden. Jedoch verschieben sich die Texte bei der Bearbeitung häufig, dies kann zu einem ungeordneten oder sogar falschen Layout führen. Um dem entgegenzuwirken, erstellte ich für jeden einzelnen Fall ein separates PDF-Dokument und auch jeweils eigene Koordinatenpaare, damit sich jeder Fall individuell verändern lässt. Die separaten Dokumente wurden mit Hilfe von Adobe Acrobat eigenhändig erstellt, wie schon in Kapitel 2.2.2 erwähnt. Die ersten Koordinaten zu finden, funktionierte mehr oder weniger nur über Trial and Error, sobald man jedoch die ersten Kreuze und Texte richtig platziert hatte, konnte man ein Muster – also gleichmäßige Abstände, nutzen, um die restlichen Koordinaten einzutragen. In Zukunft wird diese Funktionalität sicherlich noch einmal überarbeitet, zurzeit ist dies wohl aber nicht nötig.

3. Fazit

Im Rahmen der ersten Projektarbeit wurde eine interaktive Website entwickelt. Sie dient zur Erfassung und Vereinfachung technische Parameter für Automotive-Video-Projekte. Ziel war es, aus einem statischen PDF Dokument, einer interaktive Website mit einer benutzerfreundlichen Oberfläche zu schaffen. Manuelle Eingabe und eine unübersichtliche Struktur, sollten damit der Vergangenheit angehören.

Durch den Einsatz von HTML, CSS und JavaScript – sowie Plugins innerhalb Visual Studio Code, wurde das Projekt von Grund auf entwickelt. Die Gliederung in einzelne, übersichtliche Funktionen sorgt in Zukunft für eine einfache Änderung und Erweiterung der Funktionalitäten. Funktionen wie dynamische angezeigte Felder, interaktive Dropdown-Menüs und Fehleranzeigen vereinfachen die Benutzung und beeinflussen die Nutzererfahrung positiv. Durch Bilder und Tooltips erweiterte Auswahlen, machen den Vorgang des Ausfüllens auch für fachfremde Personen möglich und für alle Nutzer*innen effizienter.

Trotz fehlender Vorkenntnisse in Webentwicklung, den benutzten Sprachen und Werkzeugen, konnten der Großteil der Funktionen umgesetzt werden. Die zukünftige Bearbeitung stellt keine Herausforderung mehr dar und hält viele Möglichkeiten offen.

Zusammenfassend kann man sagen, dass die Aufgabe bis jetzt erfüllt wurde und diese Lösung im Gegensatz zu früher deutlich moderner und benutzerfreundlich ist. Sie stellt einen wichtigen Schritt der Digitalisierung von Arbeitsabläufen und Prozessen innerhalb unserer Firma dar.

Ich freue mich darauf, das Projekt in Zukunft weiterzuentwickeln und es hoffentlich bald als festen Bestandteil der Göpel-Startseite im Einsatz zu sehen.

Quellenverzeichnis

- ATS_Parameter-Checklist_Video_Projects.pdf
- Göpel Intranet und Wiki (nicht öffentlich zugänglich)

Die Bilder stammen alle entweder aus dem Wiki oder sind Screenshots vom Code / der Website.
Über den Github Link können diese Angaben nachgeprüft werden.

Eigenständigkeitserklärung

für nicht unter Aufsicht erbrachte schriftliche Prüfungsleistungen an der Dualen Hochschule Gera-Eisenach

1. Hiermit versichere ich, dass ich die vorliegende Arbeit in allen Teilen selbstständig angefertigt und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel benutzt habe. Die aus fremden Quellen (oder auch aus eigenen älteren Quellen) wörtlich oder sinngemäß übernommenen Textstellen, Gedankengänge, Konzepte, Grafiken etc. in meinen Ausführungen habe ich als solche eindeutig gekennzeichnet und mit vollständigen Verweisen auf die jeweilige Quelle versehen.

2. Meine Eigenständigkeitserklärung bezieht sich auch auf generative KI-Anwendungen (nachfolgend „generative KI"). Die umseitigen „**Vorgaben für die Studierenden der Dualen Hochschule Gera-Eisenach bei Nutzung von generativer KI für die Erstellung nicht beaufsichtigter schriftlicher Prüfungsleistungen**" habe ich zur Kenntnis genommen und bei der Erstellung der Arbeit im Falle der Nutzung generativer KI eingehalten.

3. Ich versichere des Weiteren, dass ich die vorliegende Arbeit bei keiner anderen Prüfung vorgelegt habe.

4. Mir ist bekannt, dass ein Verstoß gegen die vorgenannten Punkte prüfungsrechtliche Konsequenzen haben und insbesondere dazu führen kann, dass meine Prüfungsleistung als Täuschungsversuch und damit als „nicht bestanden" bewertet wird.

_____ _____

Ort, Datum Unterschrift