Hochschule Karlsruhe
University of
Applied Sciences

Fakultät für
Maschinenbau und
Mechatronik

Projektarbeit

Programmierung einer Webanwendung zur Evaluierung der Kritikalität von Automotive Cybervorfällen

Jan Niklas Rother, Juliane Kerpe

Bearbeitungszeitraum:	14.10.2022 – 31.03.2023
Erstkorrektor:	Prof. Dr.-Ing. Reiner Kriesten
Zweitkorrektor:	Robin Bolz

Erklärung

Wir versichern hiermit wahrheitsgemäß, die vorliegende Projektarbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles einzeln kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde.

Das Thema der vorgelegten Arbeit wurde gemeinsam von Frau Juliane Kerpe und Herr Jan Niklas Rother im Rahmen des Forschungs- und Entwicklungsprojektes bearbeitet.

Kuantan, Malaysia, den 27. März 2023

Unterschriften:

Jan Niklas Rother

Juliane Kerpe

Kurzfassung

Ziel der Projektarbeit war es, ein bestehendes Excel-Tool für die Berechnung der Grace Period im Automotive Bereich in eine Webanwendung zu überführen.

Zunächst wurden in der Konzeptions-Phase Anforderungen an die Webanwendung formuliert. In der darauffolgenden Design-Phase wurde die Software-Architektur in Form von UML-Diagrammen erstellt. Außerdem wurden Design-Vorschläge für die Benutzeroberfläche erarbeitet. In der Realisierungs-Phase wurde der Berechnungsalgorithmus aus einem bereitgestellten Excel-Tool extrahiert. Nachdem React als geeignetstes Webframework ausgewählt wurde, konnte die Webanwendung implementiert werden. Hierbei wurde vor allem Wert auf die Benutzerfreundlichkeit und eine intuitive Bedienung gelegt. Des Weiteren wurde die Webanwendung dahingehend optimiert, weniger Daten zu speichern. Entscheidend war hierbei eine große Look-up-Table aus dem bereitgestellten Excel-Tool durch eine Regression zu ersetzen. Durch diese leichtgewichtigere Lösung kann der Berechnungsaufwand und Datentransfer verringert werden.

Großer Stellenwert wurde auf eine strukturierte und nachvollziehbare Implementierung gelegt. Dies erhöht die Verständlichkeit des Codes und ermöglicht eine bessere Wartbarkeit. Des Weiteren können so leichter Anpassungen und Erweiterungen der Webanwendung vorgenommen werden.

In der Einführungs-Phase wurde die Webanwendung gehostet.

Abschließend fand eine Test-Phase statt. Hierbei wurde vor allem Wert darauf gelegt, dass die Berechnung verglichen mit den Ergebnissen des Excel-Tools die richtigen Ergebnisse liefert und Falscheingaben durch den User nicht möglich sind. Außerdem wurden die anfangs formulierten Anforderungen mit dem Ergebnis abgeglichen.

Abschließend wurde eine umfangreiche Dokumentation verfasst. Dabei wurde hauptsächlich Augenmerk auf die Verständlichkeit des Codes und der Verwendung der Webanwendung gelegt. Neben der Textform wurde auch ein Erklärvideo erstellt.

Abstract

The goal of the project was to convert an existing Excel tool for calculating the grace period in the automotive sector into a web application.

First, requirements for the web application were formulated in the conception phase. In the subsequent design phase, the software architecture was created in the form of UML diagrams. In addition, design proposals for the user interface were developed. In the realization phase, the calculation algorithm was extracted from a provided Excel tool. After React was selected as the most suitable web framework, the web application could be implemented. Here, particular emphasis was placed on user-friendliness. Furthermore, the web application was optimized to store less data. The decisive factor here was to replace a large look-up table from the Excel tool provided with a regression. This more lightweight solution reduces the calculation effort and data transfer.

Great importance was attached to a structured and comprehensible implementation. This increases the comprehensibility of the code and enables better maintainability. Furthermore, adjustments and extensions to the web application can be made more easily.

During the implementation phase, the web application was hosted.

Finally, a test phase took place. Here, particular emphasis was placed on ensuring that the calculation delivered the correct results compared with the results of the Excel tool and that incorrect entries by the user were not possible. In addition, the requirements formulated at the beginning were compared with the results.

Finally, a comprehensive documentation was written. The main focus was on the comprehensibility of the code and the use of the web application. In addition to the text form, an explanatory video was also created.

Inhaltsverzeichnis

Erklärung i
Kurzfassung ii

1. Einleitung **1**

2. Stand der Technik **2**
2.1. Schätzung der Grace Period 2
2.2. Excel-Tool 4

3. Vorgehen **6**

4. Design-Phase **7**
4.1. Software-Architektur 7
4.2. Modellierung der Oberfläche 10
4.3. Auswahl des Webframeworks 13

5. Realisierungs-Phase **16**
5.1. Ergebnisse 16
5.2. Implementierung der Klassen 18
5.2.1. ValueInput 18
5.2.2. TextInput 19
5.2.3. DisplayValues 20
5.2.4. Plot 21
5.2.5. ToolTip 21
5.2.6. Header 22
5.3. Regressionen 22
5.3.1. Grobe Regression Diagramm 22
5.3.2. Genaue Regression für die Berechnung der GP 23
5.4. Funktion der Webanwendung 24
5.5. Usability 26
5.6. Orientierung im Code 27
5.6.1. Allgemeine Tipps 28
5.6.2. index.js 29
5.6.3. App.js 29

5.6.4. GracePeriodCalculation.js . . . 30
5.6.5. GracePeriodVisualization.js . . . 30
5.6.6. DisplayValues.js . . . 30
5.6.7. ValueInput.js . . . 30
5.6.8. TextInput.js . . . 31
5.6.9. ToolTipTextProvider.js . . . 31
5.6.10.functions.js . . . 31
5.6.11.ExplainationTextsProvider.js . . . 31
5.6.12.Header.js . . . 32

6. Einführungs-Phase **33**

7. Test-Phase **36**
7.1. Validierung der Berechnungsergebnisse . . . 36
7.1.1. Beschaffung der Testdaten . . . 36
7.1.2. Abgleich der Berechnungsergebnisse . . . 36
7.2. Abgleich der Anforderungen . . . 37

8. Fazit **39**

A. Anhang **40**
A.1. Zeitplan . . . 40
A.2. Anforderungen . . . 41
A.3. Quellcode . . . 43
Abbildungen . . . 44
Abkürzungen . . . 47

Literaturnachweise **49**

1.Einleitung

Fakt ist: Neu zugelassene Fahrzeuge werden zunehmend vernetzter. Was einerseits zu mehr Sicherheit bezüglich Gefahrensituationen im Straßenverkehr und Fahrkomfort sorgen kann, birgt andererseits das Risiko von Cyberangriffen. Laut des Global Automotive Cybersecurity Reports 2023 [1] gab es in den letzten Jahren einen exponentiellen Anstieg an gefundenen Schwachstellen. Im Jahr 2022 wurden mindestens 151 Schwachstellen im Automobil-Bereich entdeckt. Dies erfordert Lösungen für effizientere Problembehebungsprozesse für oft zeitkritische Schwachstellen. Dabei wird oftmals von der Grace Period gesprochen. Darunter versteht man im Allgemeinen die Zeitspanne zwischen der Entdeckung einer Schwachstelle und das Aufspielen eines Patches. Die Automotive Security Research Group (ASRG) legt diese Grace Period grundsätzlich auf 90 Tage fest [2]. Unter bestimmten Umständen kann die Grace Period verlängert werden. Um die angemessene Grace Period von Beginn an individueller und transparenter zu berechnen, hat das Institut für Energieeffiziente Mobilität IEEM im Rahmen des Projekts SecForCARs eine Formel zur Berechnung der angebrachten Grace Period entwickelt. Diese Metrik bestimmt auf Basis des Schwachstellenrisikos, sowie der Kritikalität des angestoßenen Behebungsprozesses die jeweils angemessene Grace Period für die Patch-Entwicklung. Die Formel wurde bisher nur in Form eines Excel-Tools umgesetzt. Um die Berechnung Security-IngenieurInnen auf der ganzen Welt zur Verfügung zu stellen, sollte das Excel-Tool im Rahmen dieser Projektarbeit in ein webfähiges Online-Tool überführt werden.

Das Projekt schließt explizit die Verifikation der zugrundeliegenden Berechnungen aus. Dieser Bericht soll zur Dokumentation der geleisteten Arbeit, sowie als Hilfestellung für zukünftige Projektgruppen dienen.

2.Stand der Technik

2.1. Schätzung der Grace Period

Die Bestandteile der durch das IEEM vorgeschlagenen Formeln zur Berechnung der angemessenen Grace Period werden im Folgenden zugunsten der Vollständigkeit knapp beschrieben. Das Augenmerk der Projektarbeit liegt allerdings auf der Umsetzung in einer Webanwendung, nicht auf das Nachvollziehen der zugrundeliegenden Theorie. Die Schätzung der Grace Period erfolgt laut Bolz [3] über die folgenden Parameter:

- **Vulnerability Risk Value RV**: Der Risikowert aus einer Risikomatrix
- **Scope S**: Aussage, ob Schwachstellen Auswirkungen auf Komponenten außerhalb ihres Sicherheitsbereichs haben
- **Exploit Code Maturity EM**: Wahrscheinlichkeit, dass die Schwachstelle tatsächlich angegriffen wird
- **Remediation Level RL**: Verringerung der Bedrohung im zeitlichen Verlauf
- **Report Confidence RC**: Glaubwürdigkeit, dass die Schwachstelle tatsächlich so vorhanden ist
- **Exploit Code Dissemination ED**: Berücksichtigung, ob der Exploit Code veröffentlicht wurde
- **Incident Scale ISC**: Anzahl der betroffenen Komponenten
- **Supply Chain Scale SCS**: Anzahl der Stakeholder
- **Remediation Dissemination R**: Auswirkungen des organisatorisch-strategischen Update-Managements des Verantwortlichen. Hier soll, falls erwünscht, noch ein Wert in Tagen eingegeben werden können, um den die Grace Period erhöht werden soll.
- **Certification and Approval APP**: Mögliche Verzögerungen durch Genehmigungsvorgänge. Auch hier soll eine Verlängerung in Tagen möglich sein, die der User eingeben kann.
- **Contractual Agreements CA**: Mögliche Verzögerungen durch nötige Vereinbarungen mit Kunden und Lieferanten.

- **Default Grace Period T_0**: Minimaler Zeitraum, der zum Lösen der Schwachstelle benötigt wird

Jeder beschriebene Parameter wird vom User eine zur Schwachstelle passender Antwortmöglichkeit zugeordnet. Jeder Antwortmöglichkeit ist ein Zahlenwert zugeordnet. Dies ist in Abbildung 2.1 zu sehen. Eine Ausnahme bildet die Default Grace Period, bei der der User eine spezifische Dauer in Tagen angeben muss.

Eingaben / LUT:					
RV	Medium	High	Very High		
	0,7272	0,8636	1		
S	Unchanged	Changed			
	0	1			
EM	Not Def.	High	Functional	Proof-of-Concept	Unproven
	1	1	0,97	0,94	0,91
RL	Not Def.	Unavailable	Workaround	Temporary Fix	Official Fix
	1	1	0,97	0,96	0,95
RC	Not Def.	Confirmed	Reasonable	Unknown	
	1	1	0,96	0,92	
ED	Unclear	rnal - Black ma	External - PoC	External - not public	
	1	1	0,96	0,91	
ISC	Unclear	High	Medium	Low	
	1	1	0,97	0,95	
SCS	Low	Medium	High	Very High	
	0	0,037057	0,074113	0,11117	
R	None R=	No access req. R=	Immediately R=	not determined	determined Y=
	0	0	0	R=0,11117	Zeitwert Y
APP	None	Extension	Renewal		
	0	5 Woche	Zeitwert		
CA	None	CAL1	CAL2	CAL3	CAL4
	0	0	0	0,693	0,11117

Abbildung 2.1.: Mögliche Antwortmöglichkeiten und Werte der Parameter

Aus den Werten werden der Process-Criticality Factor *PCF* und der Time-Criticality Factor *TCF* nach Formeln von Bolz [3] berechnet. Diese Faktoren bemessen die Kritikalität der Schwachstelle bezüglich der Zeit und des Prozesses. Der PCF wird folgendermaßen berechnet:

$$PCF = SCS + (r_2 * R)$$

Der TCF kann durch diese Formel ermittelt werden:

$$TCF = RV * EM * RL * RC * ED * ISC$$

Aus diesen Faktoren kann der Grace Period Factor berechnet werden:

$$GP = (1 + PCF) * (2 - TCF)$$

Falls der Scope nicht verändert wird, muss vom Grace Period Factor noch ein Korrekturwert abgezogen werden. Die Grace Period wird basierend auf den vorigen Werten berechnet:

$$T_{\mathrm{GP}} = T_0 * GP + APP + (r * R)$$

2.2. Excel-Tool

Die im vorigen Kapitel beschriebene Berechnung der Grace Period erfolgte bisher mittels eines Excel-Tools, welches durch ein onlinefähiges Tool ersetzt werden soll.
Abbildung 2.2 zeigt einen Überblick über das bisher verwendete Excel-Tool.

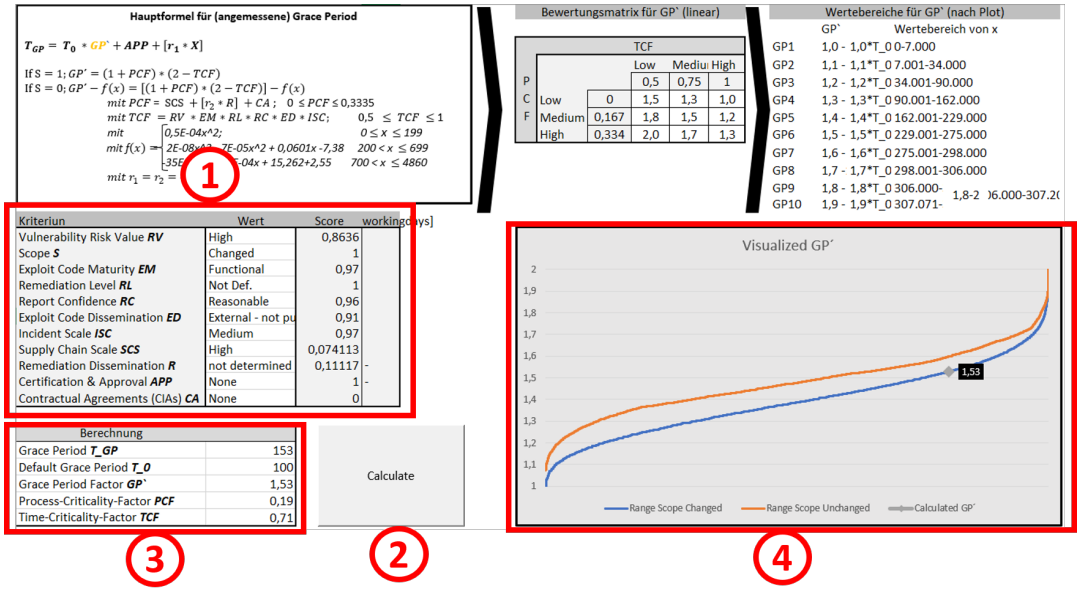

Abbildung 2.2.: Excel-Tool mit (1) User-Eingaben, (2) Berechnung-Button, (3) Ergebnissen und (4) Diagramm

Das Excel-Tool war mit einem VBA-Makro programmiert. Der User kann Eingaben über Dropdown-Menüs eingeben. Dies ist in Abbildung 2.2 in Bereich (1) zu sehen. Durch verschachtelte Wenn-Dann-Formeln wird dem ausgewählten Item ein Wert zugeordnet. Drückt der User den Berechnen-Button in Abbildung 2.2 (2), wird das hinterlegte VBA-Makro zur Berechnung ausgeführt.
Andere Angaben, wie die Default Grace Period, werden über Messageboxen realisiert. Manche Auswahlmöglichkeiten bei den Parametern CA und APP erfordern zusätzliche User-Eingaben, welche ebenfalls über Messageboxen getätigt werden müssen. Die Ergebnisse sind in Abbildung 2.2 (3) abgebildet. Zum Einschätzen der Schwere der Grace Period ist in Abbildung 2.2 (4) ein Diagramm integriert. Um im Diagramm das aktuelle Wertepaar

einzuzeichnen, wird eine Look-Up-Table LUT mit über 300000 Werten verwendet. Ein Ausschnitt davon ist in Abbildung 2.3 zu sehen. In der LUT sind alle möglichen Kombinationswerte zu finden. Im Makro wird die Zelle gesucht, deren Zellwert dem Berechnungswert des Grace Period Factors entspricht. In die Zeile wird der berechnete Wert eingetragen, sodass der Datenpunkt im Diagramm abgebildet werden kann. Durch die große Menge an Datenpunkten dauert die Berechnung gegebenenfalls mehrere Sekunden.

GP´		Differenz durch Scopechange
unchanged	changed	Absolut
1,0745872	1	-0,0745872
1,0745872	1	-0,0745872
1,0745872	1	-0,0745872
1,0745872	1	-0,0745872
1,0745872	1	-0,0745872
1,0745872	1	-0,0745872
1,0745872	1	-0,0745872
1,0745872	1	-0,0745872
1,0745872	1	-0,0745872
1,074661157	1	-0,074661157
1,074661157	1	-0,074661157
1,074661157	1	-0,074661157
1,074661157	1	-0,074661157
1,074661157	1	-0,074661157
1,074661157	1	-0,074661157
1,074661157	1	-0,074661157
1,074661157	1	-0,074661157
1,074661157	1	-0,074661157
1,074661157	1	-0,074661157
1,074735082	1	-0,074735082
1,074735082	1	-0,074735082

Abbildung 2.3.: Auszug aus LUT des Excel-Tools

3. Vorgehen

Dieses Kapitel soll einen Überblick über das Vorgehen geben und wo die Bearbeitung der einzelnen Schritte dokumentiert ist. Beim Vorgehen wurde sich am Basismodell für die Softwareentwicklung nach Brandt-Pook und Kollmeier [4] orientiert. Dabei wird die Entwicklung in folgende Phasen eingeteilt:

1. Auftragsklärungs-Phase: In dieser Phase wurde mit dem Betreuer des Projektes die genaue Aufgabenstellung geklärt. Diese Phase ist im Bericht nicht näher erläutert.

2. Konzeptions-Phase: Die Anforderungen wurden definiert und möglichst eindeutig und messbar formuliert. Diese sind in Kapitel A.2 zu finden. Zudem wurde ein Zeitplan erstellt, welcher im Laufe des Projekts aktualisiert wurde. Der Zeitplan ist in im Anhang A.1 zu finden.

3. Design-Phase: Die Softwarearchitektur und das erwünschte Verhalten der Webanwendung wurde durch passende UML-Diagramme veranschaulicht. Zum Design der Oberfläche wurden Mock-Ups erstellt. Außerdem wurde basierend auf den definierten Anforderungen aus der vorigen Phase ein Webframework ausgewählt.
   Kapitel 4 beschreibt das Vorgehen in der Design-Phase.

4. Realisierungs-Phase: Es fand das schrittweise Nachbilden des Excel-Tools basierend auf den UML-Diagrammen und den Mock-Ups statt. Teile der Berechnung wurden in Form von Regressionen durchgeführt, um die Webanwendung hinsichtlich der Leistung zu verbessern. Die Realisierungsphase ist in Kapitel 5 dokumentiert.

5. Einführungs-Phase: Die Webanwendung wurde über github.com gehostet, was in Kapitel 6 beschrieben ist.

6. Test-Phase: Aus dem Excel-Tool wurden Testdaten generiert. Diese wurden mit den Daten der Webanwendung abgeglichen. Außerdem findet ein Abgleich der Anforderungen statt. Diese letzte Phase ist in Kapitel 7 dokumentiert.

4.Design-Phase

In der Design-Phase wird die Vorarbeit zur Implementierung geleistet. Es wurde die Software-Architektur in Form von UML-Diagrammen entwickelt. Zur Modellierung der Oberfläche wurden Mock-Ups erstellt. Zuletzt wurde das geeignetste Webframework ausgewählt.

4.1. Software-Architektur

Um die Software-Architektur zu entwerfen, wurden UML-Diagramme erstellt. Um den Lösungsraum nicht einzuschränken, wurde darauf Wert gelegt, dass die Bezeichnung der Klassen möglichst abstrakt und dennoch eindeutig beschrieben wurde.

Zunächst wurde ein UML Kompositionsstrukturdiagramm erstellt. Dieses zeigt die prinzipielle Komposition der App und ist in Abbildung 4.1 zu sehen.

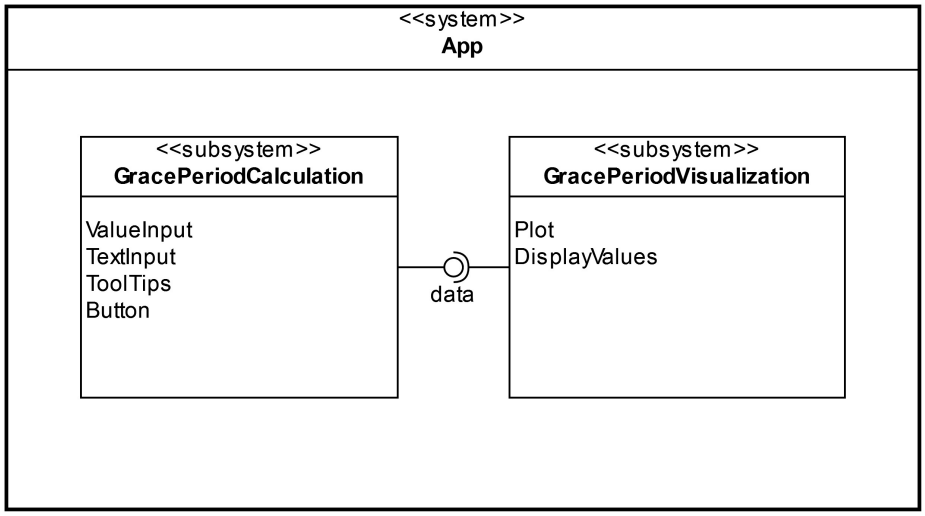

Abbildung 4.1.: UML Kompositionsstrukturdiagramm

Die App besteht im Wesentlichen aus zwei Subsystemen. Das Subsystem *GracePeriod-Calculation* ermöglicht es dem Anwender, die im Kapitel 2.1 beschrieben Werte einzugeben. Für manche Parameter sollen nur definierte Werte eingegeben werden können. Diese Eingaben sollen über einen sogenannten *ValueInput* realisiert werden. Außerdem soll die Möglichkeit bestehen, dass Zahlenwerte direkt eingegeben werden können. Diese Werte sollen über einen *TextInput* eingeben werden. Dem User soll darüber hinaus eine Hilfestellung gegeben werden, was der Werte bedeutet. Die Hilfestellung soll dem Anwender

auch bei der Auswahl des geeignetsten Wertes helfen. Somit sollen noch *Tooltips* integriert werden. Nachdem die Werte eingeben wurden, soll der Benutzer über einen *Button* die Berechnung starten können.

Die Ergebnisse der Berechnung sollen dem Anwender dargestellt werden. Diese Aufgabe soll das zweite Subsystem, die *GracePeriodVisualization*, übernehmen. Nachdem der User die Berechnung in der Komponente *GracePeriodCalculation* angestoßen hat, werden die Ergebnisse über das Interface *data* der *GracePeriodVisualization*-Komponente bereitgestellt. Über *DisplayValues* werden dem Anwender dann die Ergebnisse ausgegeben. Des Weiteren soll über einen *Plot* die Kritikalität eingeordnet und visualisiert werden.

Um die Implementierung zu erleichtern, wurde auf Grundlage des Kompositionsstrukturdiagramms ein Klassendiagramm abgeleitet. Das Klassendiagramm ist in Abbildung 4.2 zu sehen.

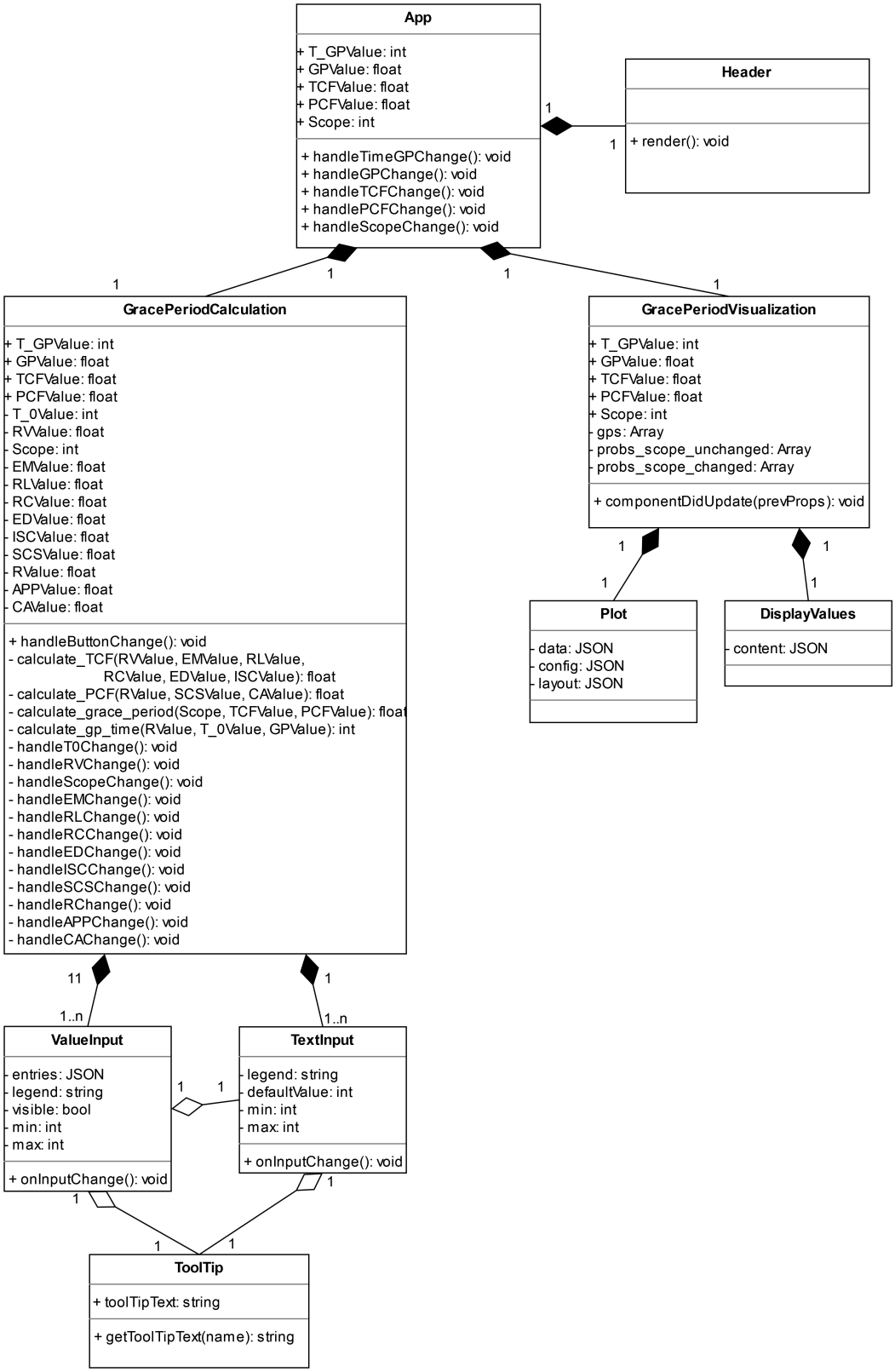

Abbildung 4.2.: UML Klassendiagramm

Das Klassendiagramm beschreibt im Wesentlichen, welche Attribute und Methoden die jeweiligen Klassen besitzen. Des Weiteren beschreibt es die Relationen der einzelnen Klassen untereinander. Die Klasse *App* besitzt die Attribute für die zu berechnenden Werte T_{GP}, *GP*, *PCF* und *TCF*. Es gibt ein Attribut für den *Scope*. Außerdem besitzt die Klasse je eine Methode, die die Änderungen verarbeitet. Im Falle des *Scope*, wenn ein neuer Wert eingegeben wird und im Falle der anderen Attribute, wenn die *GracePeriodCalculation* neu berechnete Werte bereitstellt. Die *App* ist eine Komposition aus genau einer Instanz der *GracePeriodCalculation* und genau einer Instanz der *GracePeriodVisualization*.

Zum einen besitzt die Klasse *GracePeriodCalculation* Attribute für jeden Input-Parameter, den der User eingeben kann. Für diese Attribute braucht die *App*-Klasse keinen Zugriff. Zum Anderen gibt es Attribute für die Ergebnisse der *GracePeriodCalculation*. Diese sollen später der *App* bereitgestellt werden und werden als public deklariert.

Die *GracePeriodCalculation* ist eine Komposition aus den benötigten *ValueInput*s und *TextInput*s.

Die Klasse *ValueInput* soll verwendet werden, wenn der Anwender zwischen gegebenen Werten auswählen soll. Das Attribut *entries* soll als JSON-Objekt übergeben werden, da jedem auswählbaren Begriff ein eindeutiger Zahlenwert zugeordnet ist. Außerdem benötigt jede Eingabe eine *legend*, die dem User anzeigt, um welchen Parameter es sich handelt. Des Weiteren besitzt die Klasse ein Attribut *visible*. Dieses Attribut soll es ermöglichen, zusätzlich zur Auswahl eines Parameters eine Eingabe in Textform zu tätigen. Wie in Kapitel 2.2 beschrieben, ist für manche Parameter sowohl eine Auswahlmöglichkeit, als auch eine Texteingabe nötig. Für diese Aggregation kann die Klasse *TextInput* verwendet werden.

Die Klasse *TextInput* besitzt auch eine beschreibende *legend*. Außerdem soll sie einen *defaultValue* haben. Die Attribute *min* und *max* sollen den Wertebereich einschränken, den der User eingeben kann.

Sowohl die Klasse *ValueInput* als auch die Klasse *TextInput* sollen eine Instanz der Klasse *ToolTip* aggregieren. Diese Klasse soll sich über eine Methode den jeweiligen Hilfstext für den jeweiligen Parameter holen und ihn in dem Attribut *ToolTipText* speichern.

4.2. Modellierung der Oberfläche

Um ein prinzipielles Bild von Oberfläche der Webanwendung zu bekommen, wurde ein Mock-Up erstellt. Abbildung 4.3 zeigt einen Aufbau der Benutzeroberfläche. In Anlehnung an das Kompositionsstrukturdiagramm (Abbildung 4.1) besteht das Mock-Up aus zwei Teilen. Im Bereich *Data Input* wird die *GracePeriodCalculation* instanziiert und im Bereich

Calculation Results wird eine Instanz der *GracePeriodVisualization* angelegt. Aufbauend auf das Klassendiagramm (Abbildung 4.2) ist der *Data Input* eine Komposition aus *ValueInput*s und *TextInput*s. Für das Mock-Up wurden für die *ValueInput*s Dropdown-Menüs angenommen. Die schlussendliche Umsetzung der *ValueInput*s kann sich aber im Laufe des Projekts noch ändern. Des Weiteren können sich die Anordnung und die Gestaltung in der Realisierung sich im Vergleich zum Mock-Up noch ändern. Für den Aufbau der *Calculation Results*-Spalte konnte ebenfalls das Klassendiagramm aus Abbildung 4.2 herangezogen werden. Im oberen Bereich sollen dem User die Zahlenwerte bereitgestellt werden und im unteren Bereich soll ein Plot das Ergebnis der *Grace Period* visualisieren.

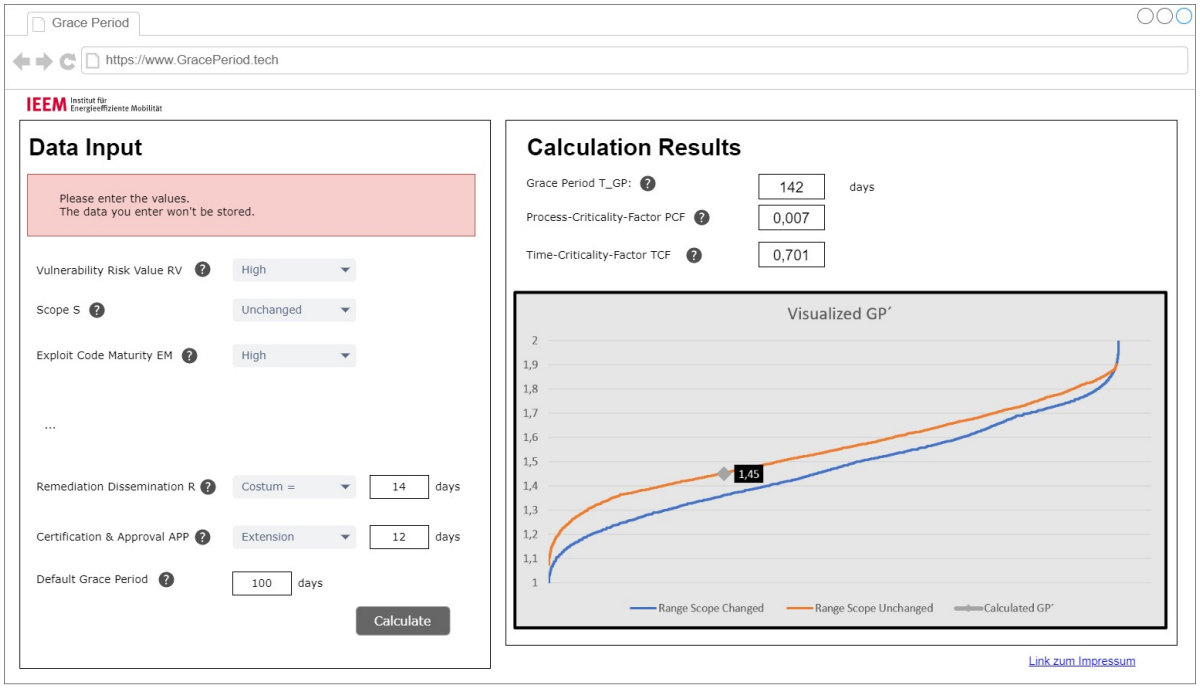

Abbildung 4.3.: Mockup 1

Das Mock-Up war außerdem sehr hilfreich, um die Projektziele während des Entwicklungsprozesses mit unserem Betreuer abzugleichen.

Nachfolgend soll auf einige Punkte eingegangen werden, die in der Webanwendung realisiert werden sollen. Im Vergleich zwischen Mock-Up 1 in Abbildung 4.3 und Mock-Up 2 in Abbildung 4.4 ist anhand des Parameters *Certification & Approval APP* beispielhaft zu erkennen, dass Texteingabefelder nur erscheinen sollen, wenn der dementsprechende Parameter beim *ValueInput* ausgewählt ist.

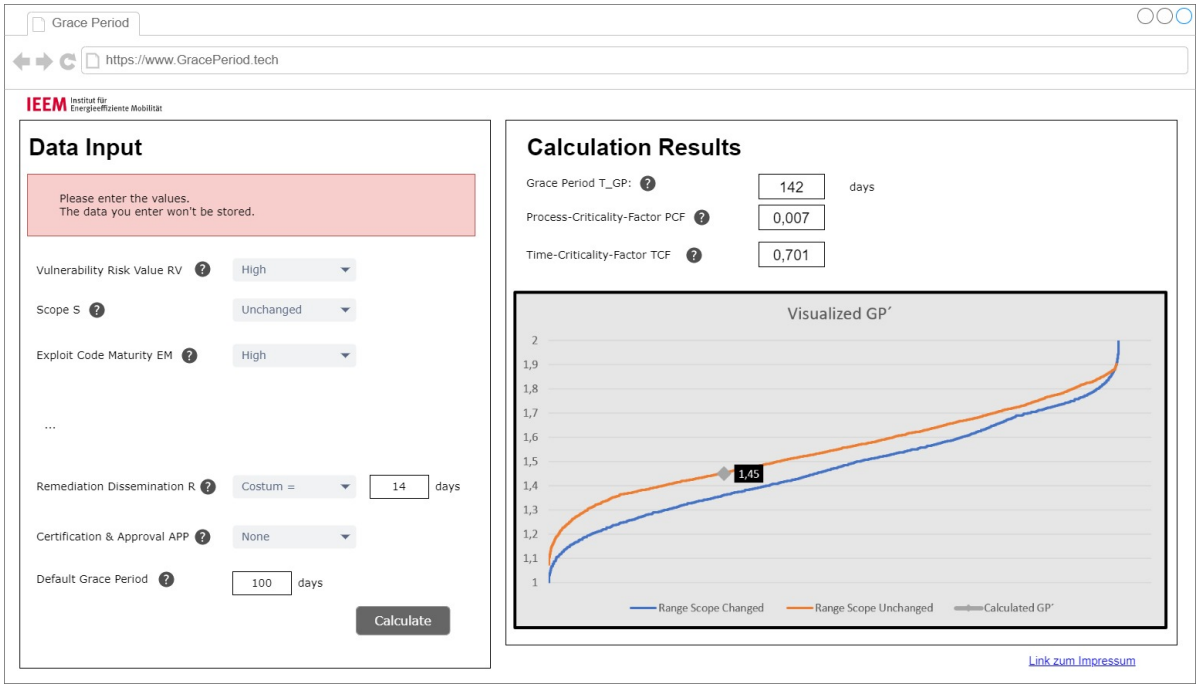

Abbildung 4.4.: Mockup 2

Abbildung 4.5 stellt dar, wie ein *ToolTip* aussehen könnte.

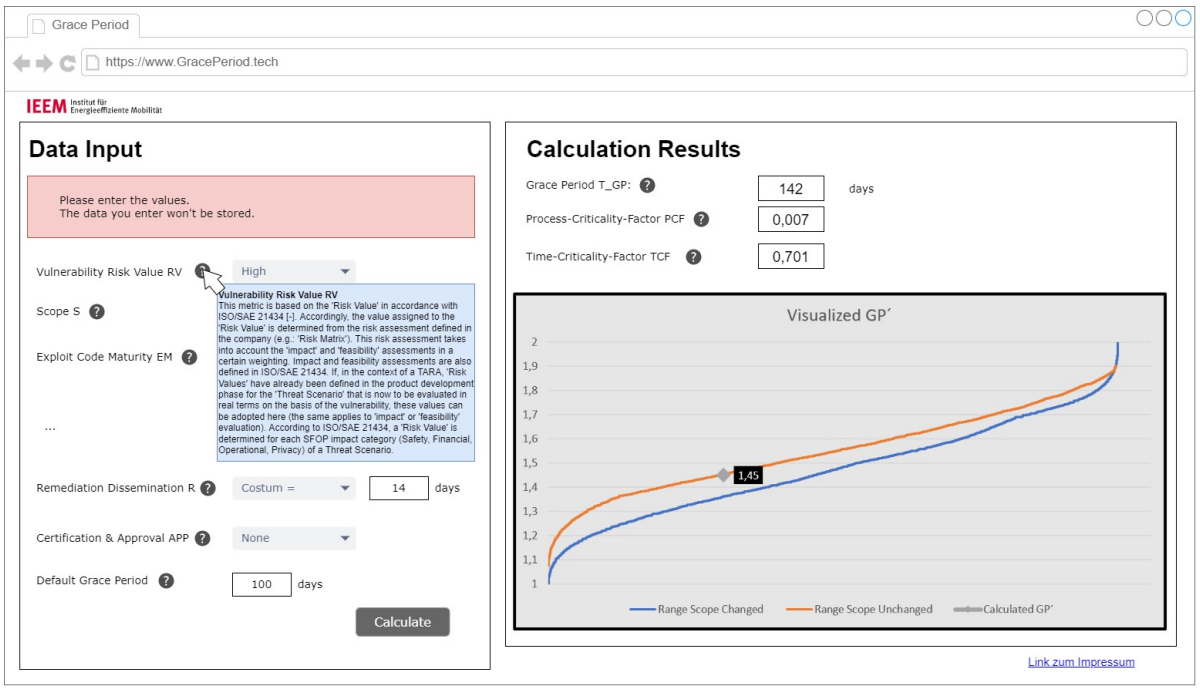

Abbildung 4.5.: Mockup 3

Wenn der User die Maus über ein Icon bewegt, erscheint ein Text, der ihm eine Hilfestellung bietet, um den geeigneten Parameter auszuwählen. Für die weitere Entwicklung ist es deshalb wichtig, diese *ToolTip*-Texte prägnant zu formulieren und gegebenenfalls Beispiele für die Werte anzugeben. Somit wird der User bei der Auswahl optimal unterstützt.

4.3. Auswahl des Webframeworks

Im nächsten Schritt der Entwicklung musste ein Framework ausgewählt werden, mit dem die Webanwendung implementiert werden soll. Abbildung 4.6 zeigt die Entscheidungsmatrix, die für die Wahl des am besten geeigneten Frameworks erstellt wurde.

	Gewich-tung	React	Vue	Angular	Svelte	JQuery	BackbBone
Summe / Ergebnis:	74	93,51%	85,27%	89,86%	81,49%	79,59%	90,81%
Einfach zu erweitern	10	10	7	9	7	8	8
Single-Page-Anwendungen unterstützt	9	10	10	10	10	10	10
Wiederverwendbarkeit der Komponenten	8	10	9	10	9	7	9
Einfach zu erlernen	6	8	10	7	9	10	10
Größe der Community	5	10	7	9	7	9	9
Effizienz	7	9	10	8	9	7	10
Performance	8	9	10	10	10	6	9
Support	8	10	8	8	6	8	10
Testbarkeit	7	7	6	10	7	7	7
Mögliche Back End Erweiterung	6	10	8	8	7	8	9

Abbildung 4.6.: Entscheidungsmatrix zum Webframework

Im Folgenden wird auf die wichtigsten Punkte der Entscheidungsmatrix eingegangen. Diese Punkte können anhand einer hohen Gewichtung in Spalte 2 *Gewichtung* identifiziert werden. Der wichtigste Entscheidungspunkt war eine einfache Erweiterung der Webanwendung. Da es sich um ein aktuelles Forschungsthema handelt, können immer neue Erkenntnisse hinzukommen. Die neuen Erkenntnisse sollen einfach in die bestehende Anwendung eingepflegt werden können.

Des Weiteren sollte das Framework Single-Page-Anwendungen SPA unterstützen. Bei einer SPA wird im Gegensatz zu einer Multi-Page-Anwendung MPA die Seite am Anfang einmal komplett geladen. Die SPA bietet daher im Gegensatz zur MPA den Vorteil, dass bei einem Klick nur ein Teil der Seite aktualisiert werden muss und nicht wie bei der MPA eine komplett neue Seite geladen werden muss. Daraus ergibt sich der Vorteil, dass die SPA nur einmal im Browser geladen werden muss und somit die Serverressourcen geschont werden. Gleichzeitig ergibt sich der Nachteil einer längeren anfänglichen Ladezeit. [5, S. 75-77] Da davon auszugehen ist, dass keine größeren Datenmengen für die Erstellung der Webanwendung notwendig sind, relativiert sich dieser Nachteil. Ein weiterer Vorteil, welcher sich aus der SPA ergibt, ist das Debugging. Wenn ein Fehler auftritt, kann der im Browser gerenderte Code einfach untersucht werden. Gängige Browser besitzen ein Untersuchungs-Tool. Es muss nicht umständlich im serverseitigen Code nach Fehler gesucht werden, sondern kann direkt im Browser gefunden werden. Ein weiterer entscheidender Vorteil der SPA ist die Möglichkeit, diese mit geringem Aufwand in eine mobile App umzubauen. Da dies zwar keine Forderung in der Projektbeschreibung war, aber perspektivisch eine interessante Möglichkeit ist, soll dieser Pluspunkt hier aufgeführt werden.

Der letzte wesentliche Punkt, auf den eingegangen werden soll, ist der Support. Für diese Projektarbeit musste eine neue Programmiersprache erlernt und ein neues Framework ver-

wendet werden. Daher war ein entscheidender Aspekt, dass es einen guten Support für das Framework gibt. Für die Ermittlung dieses Parameters wurden repräsentative Befragungen von Web-Entwicklern gesichtet und die Anzahl an Treffern auf `https://stackoverflow.com/` ausgewertet. Ein guter Support ist außerdem für Studierende hilfreich, die diese Webanwendung eventuell in Zukunft erweitern sollen.

Basierend auf allen Faktoren, wurde sich für das Webframework React entschieden. Dabei handelt es sich um eine effiziente und flexible JavaScript-Bibliothek zum Erstellen von Benutzeroberflächen.

5.Realisierungs-Phase

Zu Beginn des Kapitels zur Realisierungs-Phase werden die Ergebnisse dokumentiert. Die eigentliche Implementierung der Klassen wird im Anschluss beschrieben. Es wird beschrieben, wie die LUT aus dem Excel-Tool durch Regressionen ersetzt wurde. Zur weiteren Beschreibung der Funktion der Webanwendung dient ein UML Sequenzdiagramm. Es wird besonders auf die Aspekte zur Usability eingegangen. Zuletzt soll die Orientierung im Code eine Hilfestellung zum Verständnis des Quellcodes bieten.

5.1. Ergebnisse

Im folgenden Kapitel sollen die Ergebnisse des Projekts dargestellt werden.

Abbildung 5.1 zeigt die Webanwendung im Vollformat.

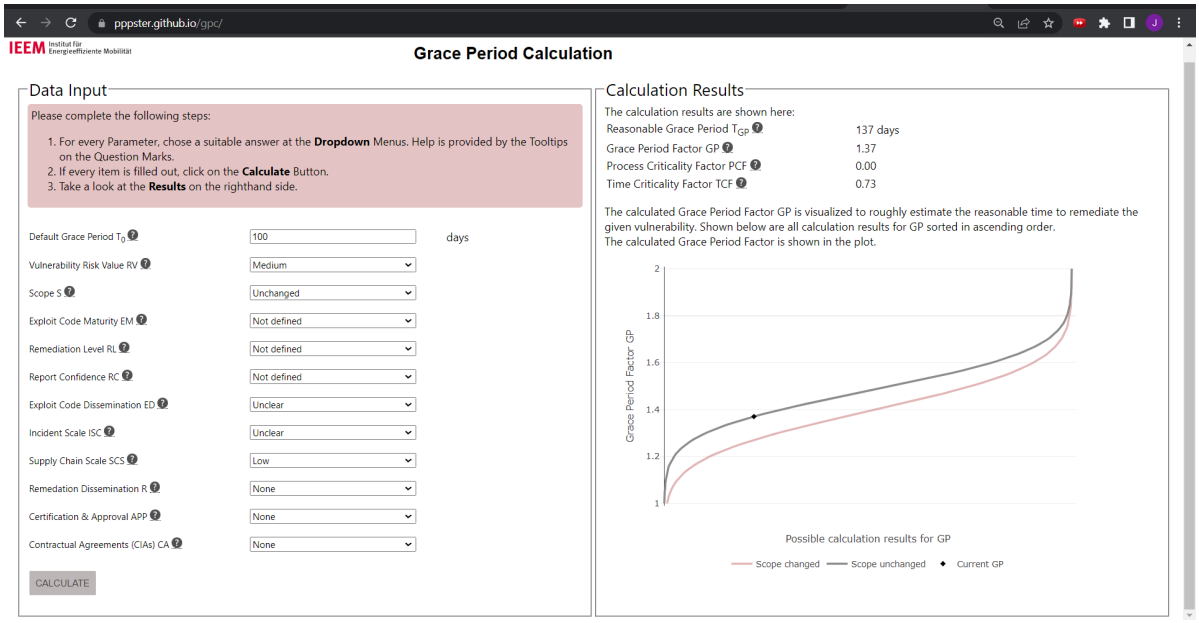

Abbildung 5.1.: Vollbild des fertigen Webframeworks

Wie im Kompositionsstrukturdiagramm (Abbildung 4.1) im Kapitel 4.1 beschrieben, besteht die Webanwendung im Wesentlich aus zwei Komponenten. Der linke Teil dient als *Data Input*. Hier kann der Anwender die passenden Werte auswählen. Des Weiteren wird im Klassendiagramm in Abbildung 4.2 dargestellt, dass die *GracePeriodCalculation* aus *TextInputs* und *ValueInputs* besteht. Die Klasse *ValueInput* instanziiert für jeden benötigten Parameter ein Dropdown-Menü, in welchem der Anwender die Auswahl an Parametern

findet. Die Klasse *TextInput* instanziiert Text-Felder, in die der Anwender den gewünschten Wert eintragen kann. Indem der Anwender den *Button* betätigt, werden die Berechnungen durchgeführt und die Ergebnisse werden an die *GracePeriodVisualization* übergeben.
Diese ist auf der rechten Seite dargestellt und dient zur Visualisierung der Ergebnisse. Die Klasse *GracePeriodVisualization* erzeugt zum einen eine *DisplayValues*-Instanz. Diese stellt dem Anwender die Ergebnisse der *GracePeriodCalculation* im oberen Bereich als Zahlenwerte dar. Zum anderen wird eine *Plot*-Instanz generiert. Diese plottet im unteren Teil der *Calculation Results*-Spalte ein Schaubild, in welchem alle Berechnungsergebnisse für GP in aufsteigender Reihenfolge aufgeführt sind. Dadurch kann der User die berechnete Grace Period grob einordnen.

Für einige Parameter ist es nötig, dass neben einem Dropdown-Menü auch die Möglichkeit besteht, einen zusätzlichen Zahlenwert einzugeben. Abbildung 5.2 zeigt dies für den Parameter *APP*.

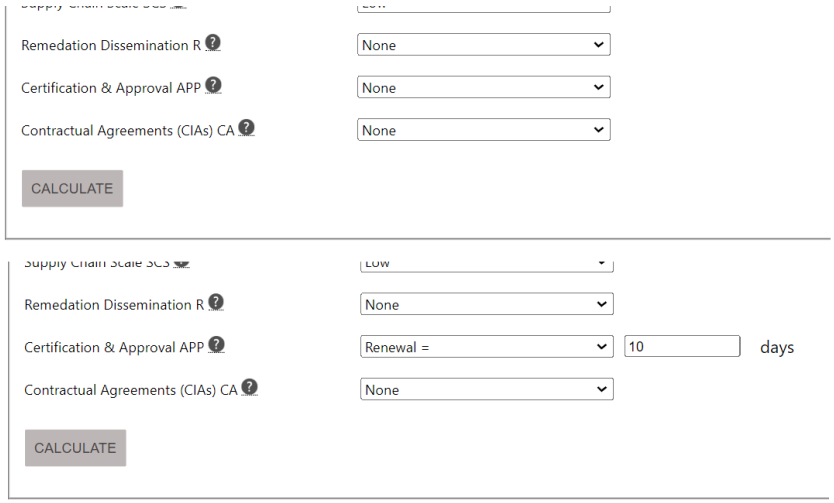

Abbildung 5.2.: oben: Textfeld zu APP verborgen / unten: Textfeld zu APP angezeigt

Wenn bei diesem Parameter z.B. der Wert Renewal eingeben wird, soll der User einen Zahlenwert für die Anzahl an Tagen eingeben. Deswegen besitzt die Klasse *ValueInput* einen Parameter *visible*, der bei Bedarf auf `true` gesetzt wird und ein Textfeld instanziiert. Hier kann der Benutzer einen angemessenen Wert eintragen.

Um den User bei der Auswahl geeigneter Input-Parameter zu unterstützen, wurden für jeden Parameter *ToolTip*s implementiert. Abbildung 5.3 zeigt einen *ToolTip*.

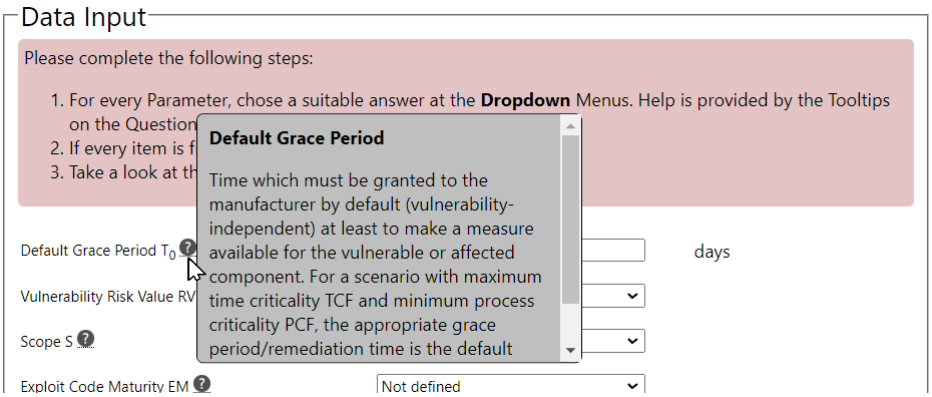

Abbildung 5.3.: Beispielhafter Tooltip

Der User kann sich den *ToolTip* durch das Bewegen der Maus über das Fragezeichen-Symbol anzeigen lassen. Da die Hilfstexte unter anderem sehr lang sein können, kann der User die Maus auf das angezeigte Textfeld bewegen und in diesem nach oben und unten scrollen.

5.2. Implementierung der Klassen

Um die Klassen aus der Konzeptionsphase umzusetzen, mussten einige grundlegende Entscheidungen getroffen werden. Dieses Unterkapitel soll dazu dienen, Entscheidungen während der Implementierung zu erläutern.

5.2.1. ValueInput

Zunächst soll nochmals verdeutlicht werden, was eine Instanz der Klasse *ValueInput* darstellt.

Abbildung 5.4 zeigt in dem grünen Kasten eine Instanz der Klasse *ValueInput*.
Links ist die Bezeichnung des Parameters zu finden. Daneben befindet sich das Fragezeichen-Icon, mithilfe dessen man sich den Hilfstext anzeigen kann. Darüber hinaus befindet sich daneben das Dropdown-Menü.

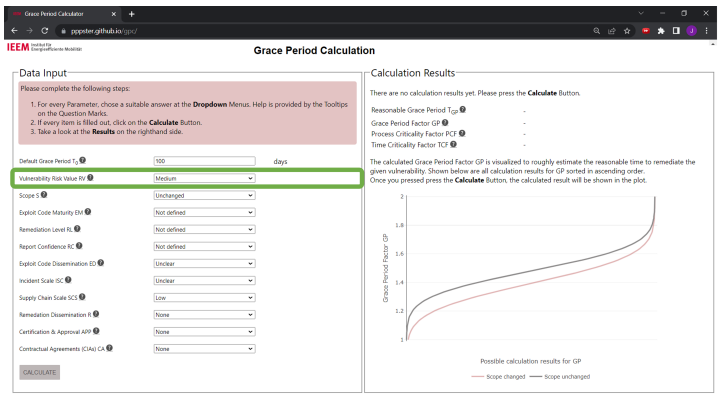

Abbildung 5.4.: Instanz der Klasse *ValueInput*

Hierzu musste sich grundlegend entschieden werden, wie die Input-Parameter ausgewählt werden sollen. Für den *ValueInput* soll pro Parameter nur exakt ein Wert ausgewählt werden können. Daher besteht die Möglichkeit, dies über Radio-Buttons oder über ein Dropdown-Menü zu realisieren. Umgesetzt wurde die Klasse *ValueInput* über ein Dropdown-Menü, da dies eine intuitive Auswahl aus einer Liste darstellt. Des Weiteren gibt es pro Parameter unterschiedliche viele Werte mit unterschiedlicher Bezeichnung, die ausgewählt werden können. Bei einer Entscheidung für Radio Buttons wäre sehr viel Text auf der Website nötig. Dies würde das Erscheinungsbild stören und die Webanwendung unübersichtlich machen. Die Idee hinter dem Dropdown-Menü ist, dass man für jeden Parameter eine ausklappbare Menge an Auswahlmöglichkeiten hat. Außerdem gibt es einen Unterschied zwischen den Anzeigetexten und den Werten. Beispielsweise erhält man beim Dropdown-Menü zum Parameter *RV* den Anzeigetext *Medium* den Wert 0,7272, *High* den Wert 0,8636 und *Very High* den Wert 1. Da der Anzeigetext für den Benutzer aussagekräftiger ist, als ein reiner Zahlenwert, wird so die Benutzerfreundlichkeit erhöht. Des Weiteren können so die *ToolTip*s kürzer gefasst werden, da die Zahlenwerte nicht erläutert werden müssen.

5.2.2. TextInput

Eine Instanz der Klasse *TextInput* ist in Abbildung 5.5 zu sehen.

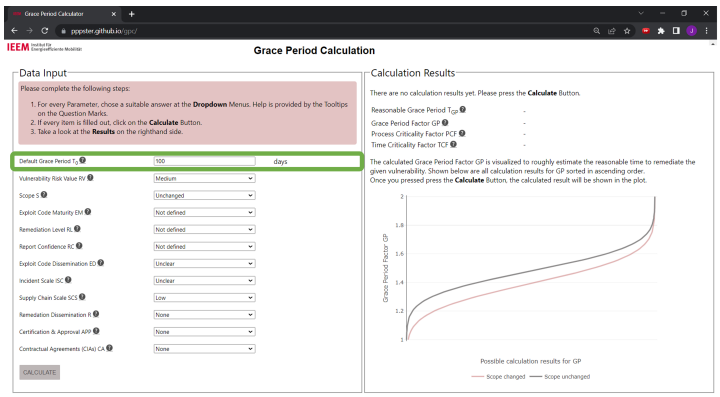

Abbildung 5.5.: Instanz der Klasse *TextInput*

Die Instanz besteht ebenfalls aus einer Beschreibung und einem Hilfstext-Icon. Der Unterschied zum *ValueInput* ist, dass das Dropdown-Menü durch ein Textfeld ersetzt wurde.

Das Textfeld der Klasse *TextInput* ist vom Typ `number`. Dieser Input-Typ ermöglicht es, ein Eingabefeld zu erzeugen, welches nur Zahlenwerte annimmt. Des Weiteren können ein Minimal- und Maximalwert angegeben werden.

5.2.3. DisplayValues

Um dem User die Zahlenwerte der *GracePeriodCalculation* bereitzustellen, werden die Werte Grace Period T_{GP}, Grace Period Factor *GP*, Process-Criticality Factor *PCF* und Time-Criticality Factor *TCF* in gerenderte HTML-Texten dargestellt. Diese Ausgabe ist in Abbildung 5.6 zu sehen.

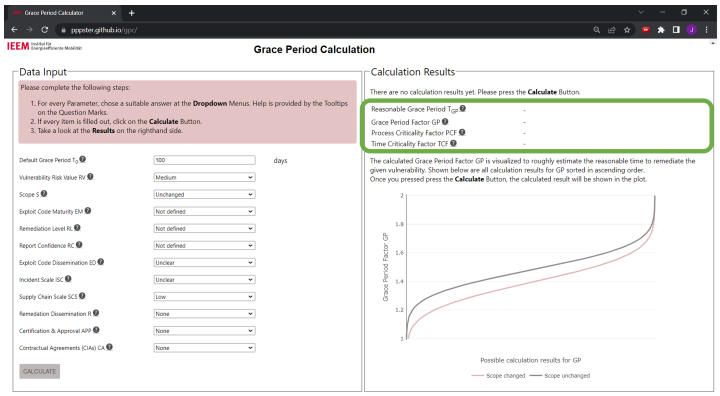

Abbildung 5.6.: Instanz der Klasse *DisplayValues*

In Abbildung 5.6 ist die tabellarische Ausgabe der berechneten Werte hervorgehoben.

5.2.4. Plot

Die *Plot*-Instanz ist in Abbildung 5.7 zu sehen. Beim Hovern über den Plot werden dem User die genauen Werte und eine Legende angezeigt. Im Plot sind zwei Graphen für die möglichen Werte mit verändertem und unverändertem Scope S zu sehen.

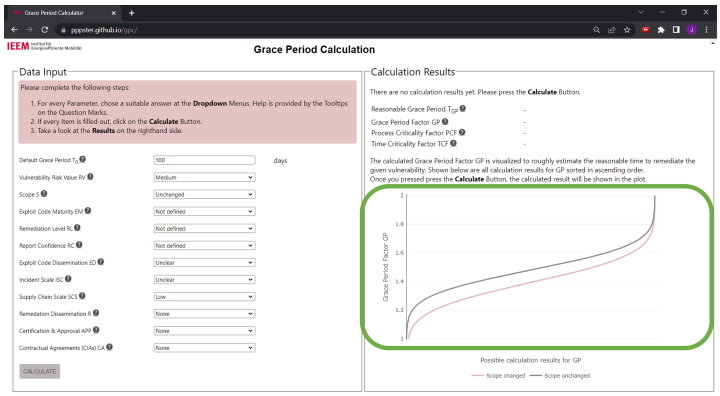

Abbildung 5.7.: Instanz der Klasse *Plot*

5.2.5. ToolTip

Die grundlegende Entscheidung für die Klasse *ToolTip* war, ob die Hilfstexte dauerhaft sichtbar sind oder nur bei Bedarf erscheinen. Da für einige Parameter die Erklärtexte sehr lang sind, wurde die Entscheidung getroffen, die Hilfstexte nur einzublenden, wenn über das Fragezeichen-Icon gehovert wird. Dieses Vorgehen hat den Vorteil, dass weniger Text auf der Seite steht und das Design schlichter und angenehmer für den User aussieht. In Abbildung 5.8 wird gezeigt, wie der Hilfstext während der Anwendung angezeigt wird.

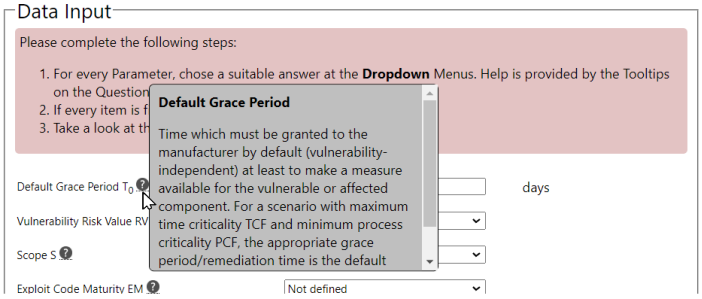

Abbildung 5.8.: Instanz der Klasse *ToolTip*

5.2.6. Header

Im Header befindet sich lediglich eine Überschrift und das Icon des Instituts für Energieeffizente Mobilität. Wenn man auf das Icon klickt, gelangt man auf die Seite des IEEM auf der HKA-Website.

5.3. Regressionen

5.3.1. Grobe Regression Diagramm

Eine LUT wie aus Kapitel 2.2 eignet sich aufgrund des erhöhten Rechenaufwands nicht für eine leichtgewichtige Webanwendung. Daher wurden die Daten zur Reduktion der Latenzzeit mittels einer Regression angenähert.
Das Diagramm dient zur groben Einschätzung des berechneten Grace Period Factors. Daher muss die Darstellung nicht allzu genau den wahren Werten aus dem Excel-Tool entsprechen. Abbildung 5.9 zeigt das anzunähernde Diagramm. Der obere Graph zeigt den unveränderten Scope und der untere den veränderten. Der schwarze Datenpunkt symbolisiert den berechneten Grace Period Factor.

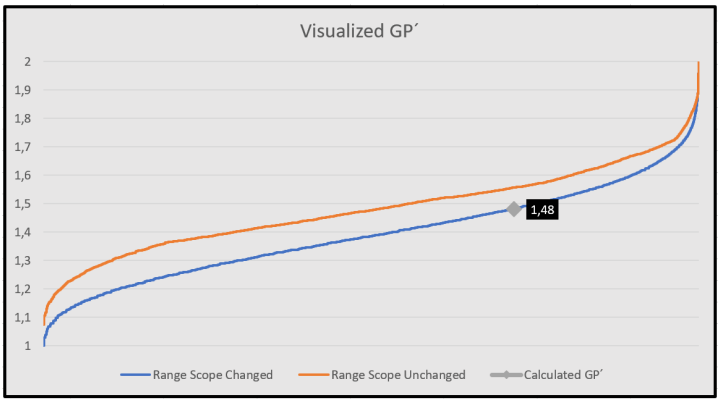

Abbildung 5.9.: Anzunäherndes Diagramm aus dem Excel-Tool

Die Werte wurden in Form eines Histogramms betrachtet. Es konnte direkt die Ähnlichkeit zu einer Normalverteilung erkannt werden.
Die gewünschte Darstellung, wie in Kapitel 2.2 gezeigt, kann als inverse Verteilungsfunktion aufgefasst werden. Dabei handelt es sich um die Umkehrfunktion der kumulierten Wahrscheinlichkeit. Da die kumulierte Wahrscheinlichkeit einfacher zu berechnen ist, wurde diese verwendet und anschließend die Werte der x- und y-Achse vertauscht. Daher wird auch im Folgenden die Ermittlung der kumulierten Wahrscheinlichkeit beschrieben. Laut

Devore [6] kann die kumulierte Wahrscheinlichkeit *CDF* eines Wertes x mit folgender Formel mithilfe der Standardabweichung μ, des Mittelwerts σ und der Fehlerfunktion *erf* berechnet werden.

$$CDF(x) = \frac{1}{2}\left(1 + \text{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right)$$

Um die Parameter Mittelwert und Standardabweichung zu ermitteln, wurden die exportierten Werte aus der LUT in Matlab eingelesen und mit der Funktion *normfit* verarbeitet.

Abbildung 5.10 zeigt den Vergleich zwischen der Wahrscheinlichkeitsverteilung des GP aus dem Excel-Tool und angenäherten Werten über eine Normalverteilung.

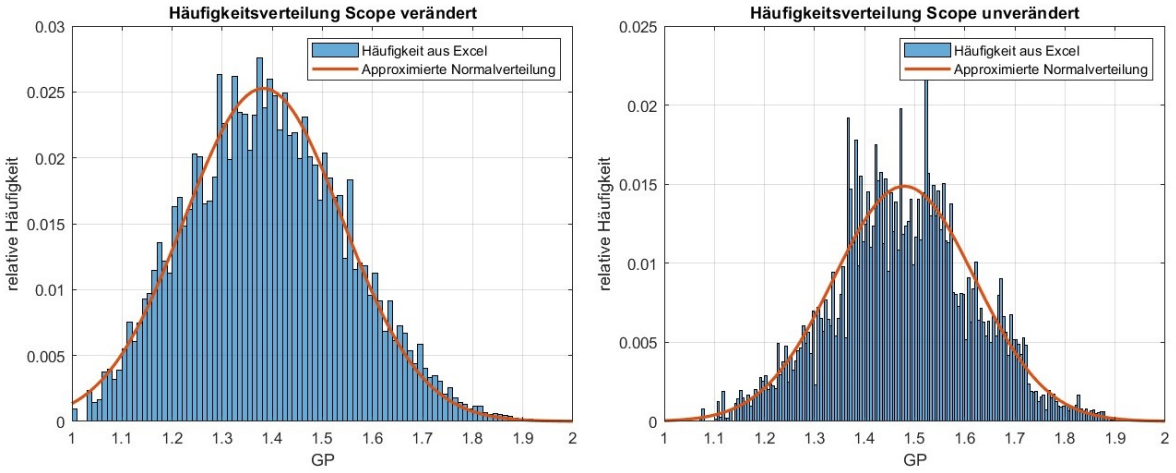

Abbildung 5.10.: Approximierte Normalverteilung mit verändertem Scope (links) und unverändertem Scope (rechts)

5.3.2. Genaue Regression für die Berechnung der GP

Während die Häufigkeitsverteilung mit verändertem Scope sehr gut mit einer einfachen CDF angenähert werden kann, ist dies bei unverändertem Scope nicht so genau möglich. Um genaue Werte für den Grace Period Factor zu berechnen, wurde sich dazu entschieden, die Abweichungen des Grace Period Factors ausgehend von der CDF mit verändertem Scope zu approximieren.

Die Abweichungen zeigen ein unstetiges Verhalten, wie in Abbildung 5.11 zu sehen ist. Daher wurde mit der Matlab-Funktion *polyfit* Polynome zweiten Grades zwischen den Unstetigkeiten approximiert. Die verschiedenen Polynome sind in Abbildung 5.11 mit verschiedenen Farben gekennzeichnet. Es wurden Polynome zweiten Grades gewählt, da Polynome ersten Grades die Krümmungen nicht abbilden könnten.

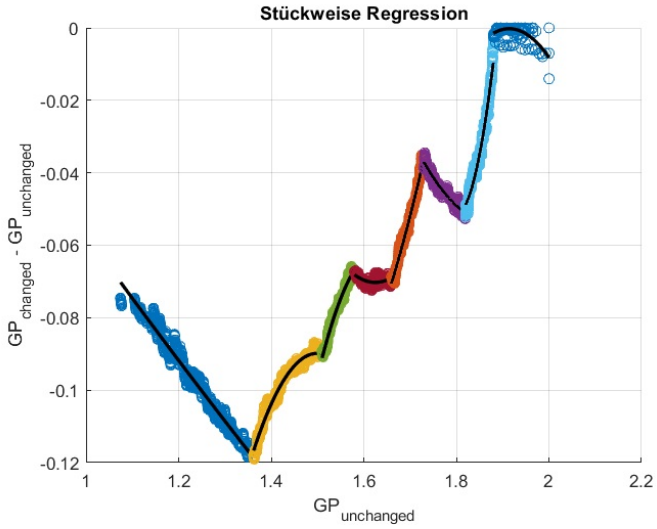

Abbildung 5.11.: Stückweise Regression

5.4. Funktion der Webanwendung

Um die Funktion des Frameworks beispielhaft zu erklären, wurde ein Sequenzdiagramm erstellt, welches in Abbildung 5.12 dargestellt ist. Es ist zu beachten, dass hierbei nur ein spezieller Fall von vielen möglichen Fällen beschrieben wird.

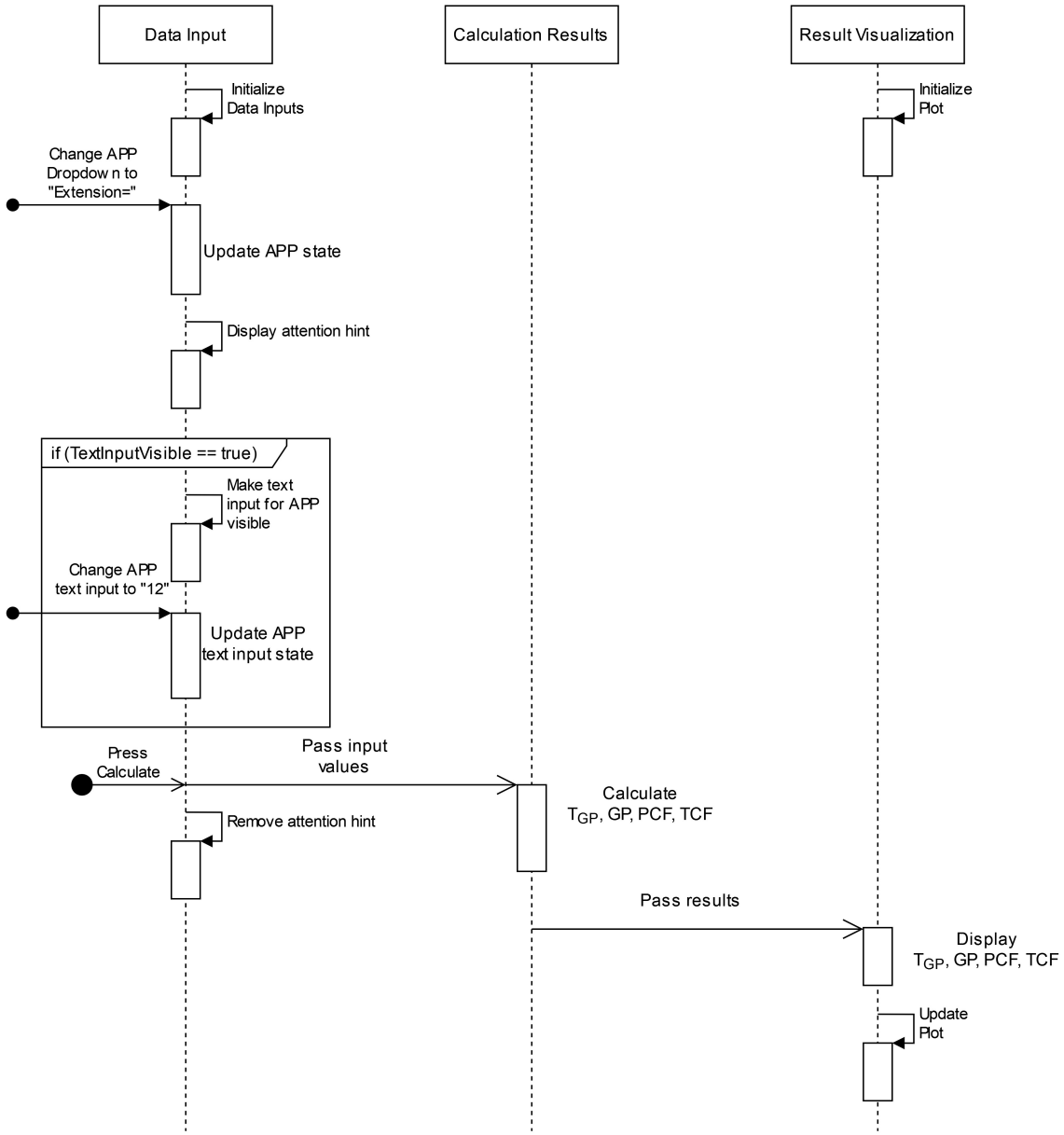

Abbildung 5.12.: UML Sequenzdiagramm

Die Abbildung 5.12 zeigt links die Lebenslinie des *Data Input*s und rechts die *Calculation Results*. Beide Komponenten werden zu Beginn initialisiert. Für den *Data Input* bedeutet dies, dass das Dropdown-Menü den ersten Wert des übergebenen JSON-Objekts annimmt und in den Textfeldern der Default-Wert ausgewählt wird. *Calculation Results* weist den User darauf hin, dass er noch keine Eingaben gemacht hat. Es gibt somit als Berechnungswerte "- - -" aus und rendert den Plot, jedoch ohne die berechnete *Grace Period* einzutragen. Nachdem die Initialisierungsphase beider Komponenten abgeschlossen ist, wird in diesem Beispiel der Parameter *APP* über das zugehörige Dropdown-Menü zu *"Extension="* geändert. Die *Data Input*-Komponente aktualisiert den Zustand des Attributs *APP*. Der Anwender wird darauf hingewiesen, dass ein Parameter verändert wurde und er den *Calculate*-Button betätigen soll, falls er die Ergebnisse mit den veränderten Parametern aktualisieren möchte. Dieser Hinweis bleibt sichtbar, bis der User den *Calculate-Button* betätigt. Da der User über das Dropdown-Menü den Wert *Extension=* ausgewählt hat, wird das Attribut *TextInputVisible* auf `true` gesetzt und die Bedingung ist erfüllt. Somit wird das Textfeld eingeblendet, über das der User den gewünschten Wert für *APP* eintragen kann. In diesem Fall gibt der Anwender den Wert 12 ein. Auch dieses Event triggert die Komponente *Data Input* den Zustand von *APP* zu aktualisieren. Wenn der Anwender nun den *Calculate*-Button betätigt, werden zum einen die aktuellen Parameter an die *Calculation Results*-Komponente weitergegeben und zum anderen wird der Hinweis, dass sich Parameter geändert haben, nicht mehr angezeigt. Die *Calculation Results*-Komponente berechnet anschließend die Werte T_{GP}, *GP*, *PCF* und *TCF*. Diese Werte werden für den Anwender visualisiert. Außerdem wird der Plot aktualisiert, indem der berechnete Wert für den Grace Period Factor *GP* geplottet wird.

5.5. Usability

Um die Anwenderfreundlichkeit zu steigern, wurden folgende Features implementiert:

1. Wenn der Anwender selbst Werte in ein Textfeld eingeben muss, wurde der Wertebereich begrenzt, um unrealistische Eingaben, wie z.B. eine negative Anzahl an Tagen, abzufangen. Wenn ein zu großer bzw. zu kleiner Wert eingegeben wird, wird der Wert auf das Maximum bzw. Minimum gesetzt.

2. Der Anwender kann nur Zahlenwerte in ein Textfeld eingeben. Werden Buchstaben oder Sonderzeichen eingeben, werden diese von der Webanwendung nicht verarbeitet.

3. Der Anwender hat mehrere Möglichkeiten, die Werte in Textfeldern zu bearbeiten. Zum einen kann er sie eingeben. Zum anderen kann er sie über die Pfeile am rechten Rand des Textfeldes verändern. Des Weiteren kann er sie mit den Pfeiltasten erhöhen oder verringern.

4. Mit der Tabulator-Taste kann der Anwender zum nächsten Eingabefeld springen. Mit der Tabulator-Taste + SHIFT kann er zum vorherigen Eingabefeld springen.

5. Wenn ein Dropdown-Menü ausgewählt ist, kann der Benutzer mit den Pfeiltasten durch die Einträge scrollen.

6. Wenn der Anwender einen beliebigen Wert im *Data Input* ändert, wird er darauf hingewiesen, dass er die Berechnung erneut durchführen sollte, um die *Calculation Results* zu aktualisieren.

5.6. Orientierung im Code

Um die Verständlichkeit des Codes und eine mögliche Weiterentwicklung des Codes zu erleichtern, werden in diesem Kapitel einige Hilfestellungen gegeben.

In Abbildung 5.13 ist die Ordnerstruktur des Projekts zu sehen.

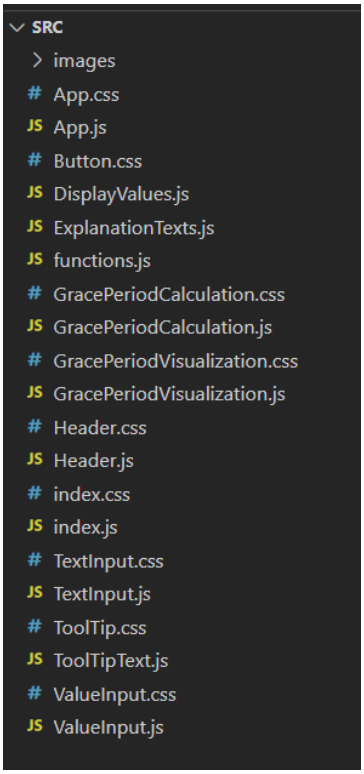

Abbildung 5.13.: Ordnerstruktur

Um nochmals die Verständlichkeit für den Quellcode zu erhöhen, werden die Hilfestellungen zu den aufgeführten Dateien nicht in alphabetischer Reihenfolge erläutert, sondern vom Grundlegenden zum Detaillierten. Zunächst sollen jedoch einige allgemeine Tipps gegeben werden, die beim Programmieren mit React hilfreich sein können.

5.6.1. Allgemeine Tipps

1. *CSS*

   Das Erscheinungsbild der Webanwendung wird über .css-Files gesteuert. Somit ist für jede Komponente, die von der Standardformatierung abweicht, ein .css-File angelegt. Diese beeinflussen auch oft die Anordnung der einzelnen Komponenten.

2. `state`

   `states` beschreiben den aktuellen Zustand der Variablen in einer Komponente. Die `states` werden über `setState`-Funktionen geändert, die meistens von `events` getriggert werden.

3. `props`

`props` beschreiben Properties. Mit diesem Key-Word kann auf die übergebenen Parameter zugegriffen werden.

4. Data-Binding

   Um Daten von einer Child-Komponente an die Parent-Komponente zu übergeben, muss dies mittels eines sogenannten bindings gemacht werden. Im Konstruktor der Parent-Komponente wird deklariert, wie die Funktion für das binding heißt. Innerhalb der Klasse wird die Funktion implementiert. Diese Funktion wird an die Child-Komponente übergeben. Wenn sich nun ein Wert in der Child-Komponente ändert, wird die übergebene binding-Funktion aufgerufen und somit der Wert an die Parent-Komponente übergeben.

5. `componentDidUpdate(prevProps)`

   Diese Funktion vergleicht die vorherigen `props` mit den aktuellen `props` und aktualisiert bei einer Änderung die `state`s.

5.6.2. index.js

In diesem File wird die *App* instanziiert. Wie im Klassendiagramm in Abbildung 4.2 beschrieben, ist die *App* eine Komposition. Um die Webanwendung im Browser zu rendern, muss in einer root-Datei die *App*-Instanz erzeugt werden. Dies passiert in dieser Datei. Hier wird auf das benötigte html-File *index.html* verwiesen.

5.6.3. App.js

In diesem File wird die *App*-Klasse implementiert. Besonderes Augenmerk ist hier auf das Data-Binding zu legen. Wie in Kapitel 5.6.1 beschrieben, werden hier die Ergebnisse der Child-Komponente *GracePeriodCalculation* an die Parent-Komponente *App* übergeben. Außerdem werden hier die JSON-Objekte angelegt, in denen die Key-Value-Paare für die Dropdown-Menüs gespeichert sind. Müssen z.B. die Werte in Zukunft angepasst oder erweitert werden, ist dies in dieser Datei vorzunehmen. Da die einzelnen JSON-Objekte für die Parameter aber erst bei der Instanziierung der *ValueInput*s in der *GracePeriodCalculation*-Komponente benötigt werden, müssen diese bereits jetzt an die *GracePeriodCalculation*-Komponente übergeben werden.

5.6.4. GracePeriodCalculation.js

In dieser Datei wird die Klasse *GracePeriodCalculation* implementiert. Es werden zunächst im Konstruktor für alle *ValueInput*s und *TextInput*s binding-Funktionen deklariert, um die ausgewählten Werte der Dropdown-Menüs und die eingegebenen Werte in die *TextInput*s für die Berechnung bereitzustellen. Des Weiteren wird der `state` für jede *ValueInput*-Komponente auf den ersten Wert des zugehörigen JSON-Objekts gesetzt. Wenn der default-Wert für die Dropdown-Menüs verändert werden soll, kann einfach der Index dementsprechend abgeändert werden.

5.6.5. GracePeriodVisualization.js

In dieser Datei befindet sich die *GracePeriodVisualization*. Zunächst werden die `state`s der Komponente auf die von der *App* übergebenen `props` gesetzt. Essenziell für die *GracePeriodVisualization* ist die `componentDidUpdate`-Funktion. Dieser Vergleich und das daraus resultierende Update der `state`s ist entscheidend, um immer das aktuellste Ergebnis der *GracePeriodCalculation* darzustellen. Es ist noch wichtig zu erwähnen, dass die *Plot*-Komponente nicht selbst implementiert wurde. Hier konnte auf das Package *react-plotly* zurückgegriffen werden. Da die Dokumentation sehr gut ist, wird hier nicht näher auf dieses Package eingegangen.

5.6.6. DisplayValues.js

Hier ist der Quellcode für die *DisplayValues*-Komponente abgelegt. Die Komponente besitzt lediglich die Attribute `explain_text` und `content`. Diese werden der *DisplayValue*-Komponente von der *GracePeriodVisualization*-Komponente übergeben.
Der `explain_text` befindet sich in den ersten Zeilen in der *Calculation Results*-Spalte. Diese ändert sich je nachdem, ob bereits eine Berechnung durchgeführt wurde. In dem Attribut `content` werden die Ergebnisse, die als Property übergeben werden, gespeichert. Beim Rendern werden diese Ergebnisse in einer Tabelle dargestellt. Auch die *DisplayValues*-Komponente besitzt eine `componentDidUpdate`-Funktion, da die Aktualität der Ergebnisse von den übergebenen `props` abhängig ist.

5.6.7. ValueInput.js

In dieser Datei wird die Klasse *ValueInput* implementiert. Das Dropdown-Menü hat die Auswahlmöglichkeiten, die mit dem `entries`-Property übergeben wird. Des Weiteren wird

über den `state selected` das erste Element des übergebenen JSON-Objekts als default-Wert angenommen. In der `handleChange`-Funktion wird abgefragt, ob der ausgewählte Wert des Dropdown-Menüs ein ¨=¨ besitzt. In diesem Fall wird das Attribut `visible` auf `true` gesetzt und das zusätzliche Textfeld erscheint.

5.6.8. TextInput.js

Die Klasse für den *TextInput* wird in diesem File implementiert. Um mögliche Fehler bei der *GracePeriodCalculation* abzufangen, werden die Properties `defaultValue`, `min` und `max` übergeben und in den zugehörigen `state`s gespeichert.

5.6.9. ToolTipTextProvider.js

In *ToolTipTextProvider.js* sind die Funktionen geschrieben, die den passenden Hilfstext abhängig von der `legend` der jeweiligen Instanz zurückgeben. Wenn in Zukunft die Hilfstexte geändert werden sollen, muss das in diesem File gemacht werden. Des Weiteren ist zu beachten, dass wenn die Legenden angepasst werden, auch die Abfrage der Legenden in dieser Datei abgeändert werden. Somit wird sichergestellt, dass der richtige *ToolTip* zurückgegeben wird.

5.6.10. functions.js

In *functions.js* sind verschieden Hilfsfunktionen implementiert. Zum einen finden sich hier die Berechnungen für die Werte Grace Period T_{GP}, Grace Period Factor *GP*, Process-Criticality Factor *PCF* und Time-Criticality Factor *TCF*. Wie bereits in Kapitel 5.3.1 aufgezeigt, wäre die LUT zu groß und wird durch eine Regression angenähert. Zum anderen werden auch die Funktion für diese Regression in diesem File implementiert.

5.6.11. ExplainationTextsProvider.js

In dieser Datei finden sich die *ExplainationTexts*. Diese Texte sind im Gegensatz zu den *ToolTips* immer in der Webanwendung zu sehen. Ein Beispiel ist das rot hinterlegte Feld in der *Data Input*-Spalte. Je nach der aktuellen Konfiguration der App können die Texte sich ändern. Sollten in Zukunft die Texte geändert werden, kann das in dieser Datei vorgenommen werden.

5.6.12. Header.js

Die *Header*-Klasse rendert bisher nur das IEEM-Logo und eine Überschrift. Des Weiteren ist das IEEM-Logo mit einem Hyperlink hinterlegt, welcher den Anwender beim Klicken auf die Seite des IEEM auf der HKA-Website bringt. Sollte der Header zukünftig erweitert werden, muss diese Klasse dementsprechend abgeändert werden.

6.Einführungs-Phase

Die Implementierung erfolgte in der Entwicklungsumgebung Visual Studio Code. Zur Entwicklung wurde die Webanwendung auf dem Local Host gelauncht. Dies lässt sich durch den Befehl `npm start` bewerkstelligen. Um die Webanwendung nach der Implementierung dem Anwender bereitzustellen, muss sie gehostet werden. Es wurde während der Projektarbeit vereinbart, dass das Hosting nicht mehr Teil der Arbeit ist. Dennoch wurde für Testzwecke ein Webhosting als Github-Page erzeugt. Diese Kapitel soll in Kürze erklären, wie man ein solches Hosting erzeugt. Im Rahmen der Projektarbeit wurde das folgende Tutorial als Grundlage genutzt: `https://github.com/gitname/react-gh-pages`. Nachfolgend wird eine Schritt-für-Schritt-Anleitung bereitgestellt:

1. Erzeugen eines Repositories

   Zunächst muss eine neues Github-Repository erzeugt werden, in welchem der Quellcode für die Webanwendung liegt.

2. `gh-pages` installieren

   Um eine React-App als Github-Page zu hosten, muss das package `gh-pages` installiert werden. Hierzu muss man in das Verzeichnis des Repositories wechseln und über eine Kommandozeileneingabe folgenden Befehl eingeben:

   ```
   npm install gh-pages –save-dev
   ```

3. Homepage-Property setzen

   Nachdem das `gh-pages`-Package installiert wurde, wird eine *package.json* Datei im Repository angelegt. Diese muss nun um ein Homepage-Property erweitert werden. Abbildung 6.1 zeigt das Homepage-Property der Webanwendung.

Abbildung 6.1.: Homepage-Property

   Es ist darauf hinzuweisen, dass es hier eine Konvention gibt, wie Github-Pages benannt werden. Sie folgen immer folgendem Schema:

`https://`[Name des Github-Accounts]`/github.io/`[Name des Repositories]

4. Einstellung für das Deployment

   Um die Webanwendung bereitzustellen, müssen Skripte für das Deployment und das Predeployment eingesetzt werden. Hierzu muss abermals die *package.json*-Datei erweitert werden. Abbildung 6.2 zeigt die zwei Zeilen Text, die in die *package.json*-Datei eingefügt werden müssen.

Abbildung 6.2.: deployment

5. Ein Remote Repository hinzufügen

   Dem lokalen Repository muss ein sogenanntes remote Repository zugewiesen werden, dessen URL auf das in Schritt 1 erstellte GitHub-Repository verweist. Hierzu muss folgende Kommandozeileneingabe im lokalen Repository ausgeführt werden:

   ```
   git remote add origin
   https://github.com/ [Github-Account]/[Repositoriy].git
   ```

6. Die App auf Github pushen

   Im letzten Schritt muss der Code in eine lauffähige Datei umgewandelt werden. Hierzu muss die folgende Kommandozeileneingabe im Repository ausgeführt werden:

   ```
   npm run deploy
   ```

   Nach wenigen Minuten ist die Github-Page dann unter der URL, die als Homepage-Property gesetzt wurde, erreichbar.

Wenn nun Änderungen an der Webanwendung vorgenommen wurden und diese auf Github gepusht wurden, muss nur der letzte Schritt wiederholt werden, um die Webanwendung zu aktualisieren.

Des Weiteren soll darauf hingewiesen werden, dass es zu Problemen mit der Darstellung des Favicons auf der Github-Page kam. Um diese Probleme zu umgehen, muss der Syntax beibehalten werden, der in der *index.html*-Datei beschrieben ist.

7. Test-Phase

Die Test-Phase gliedert sich in die Validierung der Berechnungsergebnisse und in den Abgleich der Anforderungen.

7.1. Validierung der Berechnungsergebnisse

7.1.1. Beschaffung der Testdaten

Um die mit Javascript verfassten Funktionen zu testen, mussten zunächst Testdaten aus dem Excel-Tool generiert werden. Dazu wurde ein zusätzliches VBA-Makro erstellt. Dieses schreibt pro Faktor zufällig eines der erlaubten Elemente in die Zelle mit dem Dropdown-Menü. Die zugehörigen Werte werden vom Excel-Tool selbst eingetragen. Anschließend wird das vorhandene Berechnungs-Makro aufgerufen. Dieses wurde nur in einer Hinsicht verändert: Statt Werte über eine Messagebox einzugeben, nimmt das Makro immer denselben Wert für APP und die Default Grace Period. Nachdem die Berechnung ausgeführt wurde, schreibt das neue Makro die ausgewählten Faktoren, sowie die einzelnen Berechnungsergebnisse in die nächste freie Zeile einer separaten Registerkarte. Durch automatisierten Aufruf dieses Ablaufs wurden so 300 Parameterkombinationen und deren Berechnungsergebnisse abgespeichert. Ein Ausschnitt aus den Ergebnissen ist in Abbildung 7.1 zu sehen.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	RV	S	EM	RL	RC	ED	ISC	SCS	R	APP	CA	T_GP	T_0	GP	PCF	TCF
2	0,7272	0	0,97	0,96	1	1	1	0,037057	1	0	0	166,34345	100	1,46343	0,03706	0,67717
3	0,7272	0	1	1	1	1	1	0,037057	1	0	0,11117	173,93303	100	1,53933	0,14823	0,72720
4	1	1	0,97	0,97	1	1	0,97	0	0,11117	0	0	130,82051	100	1,20821	0,11117	0,91267
5	0,7272	1	1	0,95	0,96	1	1	0,074113	1	1	0,11117	168,44787	100	1,58448	0,18528	0,66321
6	1	0	1	1	1	1	0,97	0,037057	1	0	0	134,55638	100	1,14556	0,03706	0,97000
7	0,7272	0	1	1	1	1	0,95	0,074113	1	0	0,11117	182,23724	100	1,62237	0,18528	0,69084
8	1	0	1	1	1	1	0,95	0,074113	0	0	0	132,53436	100	1,22534	0,07411	0,95000
9	0,8636	0	1	0,97	0,92	0,91	0,95	0,074113	0	1	0	152,05621	100	1,52056	0,07411	0,66625

Abbildung 7.1.: Automatisiert extrahierte Berechnungsergebnisse

Um die Parametersätze in einer JavaScript-Datei verwenden zu können, wurden sie als CSV-Datei exportiert.

7.1.2. Abgleich der Berechnungsergebnisse

Javascript und somit auch React stellen Testframeworks bereit. Im Projekt sind allerdings der Abgleich der Berechnungsergebnisse die einzigen Komponenten, die mithilfe von Unit-

Tests getestet werden können. Daher wurde sich entschiedenen, statt mittels Unit-Tests, die Berechnungsergebnisse auf eine einfachere Weise zu vergleichen.

Für der Ergebnisse der Werte für Process-Criticality Factor *PCF* und Time-Criticality Factor *TCF* ergeben sich keine Abweichung von den mit dem Excel-Tool berechneten Werten.

Für die Ergebnisse des Grace Period Factor *GP* ergeben sich kleine Abweichungen von den Ergebnissen des Excel-Tools. Diese betragen in absoluten Werten im Mittel 0,0132 und maximal 0,0247, wenn der Scope *unchanged* ist. Dies entspricht einer prozentualen Abweichung von 0,72% bzw. 1,35%. Diese Abweichungen sind auf die Umstellung der LUT auf eine Regression zurückzuführen. Dies ist im Kapitel 5.3 beschrieben.

Auch für die Ergebnisse für die Reasonable Grace Period T_{GP} ergeben sich aus dem genannten Grund Abweichungen. Diese betragen absolut im Mittel 1,32 und maximal 2,47. Dies entspricht einer prozentualen Abweichung von 0,69% bzw. 1,29%. Auch diese Abweichung tritt nur auf, wenn für den Scope *unchanged* ausgewählt wurde. Auch hier ergibt sich der Fehler aus der Annäherung der LUT durch eine Regression.

7.2. Abgleich der Anforderungen

Um den Erfolg der Projektarbeit zu bewerten, werden in diesem Kapitel die Ergebnisse mit den Anforderungen abgeglichen. Im Anhang in Kapitel A ist die Anforderungsliste zu sehen. Für jede Anforderung wurde ein Test-Case definiert. Nachfolgend soll auf die Anforderungen eingegangen werden, die vom Ergebnis abweichen. Wie in Abbildung A.3 im AnhangA gezeigt, wurden die Tests für die Anforderungen erfüllt. Drei Tests wurden nicht bzw. teilweise erfüllt. Der erste Test, der nicht restlos erfüllt wurde, war, dass die Webanwendung in den vier Browsern Google Chrome, Microsoft Edge, Safari und Mozilla Firefox funktioniert. Während des Testens ist aufgefallen, dass man in den Browsern Safari und Mozilla Firefox Buchstaben und Sonderzeichen in die Textfelder eingeben kann. Nach einer Internetrecherche wurde entdeckt, dass dies ein grundlegender Fehler von React ist. Die Quellen dafür wurden in dem Kommentar zum Test angegeben. Anzumerken ist, dass die Eingabe von Buchstaben und Sonderzeichen die Webanwendung nicht abstürzen lässt. Es werden lediglich nicht plausible Ergebnisse berechnet.
Der zweite Test, der nicht erfüllt wurde, bezieht sich auf die Sprache der Webanwendung. Als Soll-Anforderung wurde definiert, dass die Webanwendung auch in deutscher Sprache zur Verfügung stehen soll. Dieses Feature wurde aus Zeitgründen nicht implementiert, da es sich um eine Soll-Anforderung handelt. Dennoch soll ein kurzer Ausblick gegeben werden, wie diese Funktion hinzugefügt werden könnte. Hierzu würde eine weitere Schaltflä-

che benötigt werden, die der User betätigen kann, um die Sprache zu ändern. Da es die gesamte *App* betrifft, sollte die Schaltfläche auch in der *App*-Komponente instanziiert werden. Der Wert, der durch die Schaltfläche beeinflusst wird, wird dann als weiterer Parameter an die jeweiligen Funktionen in den Dateien *ToolTipTextProvider.js* und *Explaination-TextsProvider.js* übergeben, die dann basierend auf dem Parameter den Text in der jeweiligen Sprache zurückgeben.
Der letzte Test, der nur teilweise erfüllt wurde, bezieht sich auf die Berechnung, hier kommt es bei der Berechnung der Werte T_{GP} und *GP* zu geringen Abweichungen zu den Ergebnissen des Excel-Tools. Die Gründe hierfür wurden bereits im Kapitel 7.1 erläutert.

8.Fazit

Während des Projekts wurde eine vollständig funktionierende Webanwendung zur Berechnung der Grace Period für Automotive Cybervorfälle entwickelt.
Die Formeln zur Berechnung wurden vom IEEM in Form eines Excel-Tool bereitgestellt.
Durch strukturiertes Vorgehen nach Lehrbuch konnte der Terminplan eingehalten werden.
Die Anforderungen konnten bis auf Fehlformulierungen ebenfalls restlos erfüllt werden.
Die Webanwendung funktioniert in allen geforderten Browsern. Lediglich das Abfangen fehlerhafter Eingaben in die Textfelder ist aufgrund eines Fehlers von React nicht in allen Browser-Versionen möglich. Da dies die Webanwendung nicht zum Abstürzen bringt, wird dieser Abweichung zur Anforderung kein großer Wert beigemessen. Die neu entwickelte Webanwendung ist hinsichtlich einer schnellen Berechnung optimiert. So bemerkt der User kaum eine Latenzzeit zwischen Starten der Berechnung und der Bereitstellung der Ergebnisse. Dies konnte unter anderem durch das Ersetzen der Look-Up-Table aus dem Excel-Tool durch Regressionen erzielt werden. Mittels Tests konnte nachgewiesen werden, dass der Einsatz der Regressionen nur zu minimalen Abweichungen im Gegensatz zu der Berechnung mithilfe der Look-Up-Table führt.
Besonderer Wert wurde auf die Benutzerfreundlichkeit gelegt. Dies wurde unter anderem durch eine klare Trennung der Benutzereingaben und der Berechnungsergebnisse erreicht.
Die Software wurde so konzipiert, dass sie von künftigen Studierenden möglichst leicht erweiterbar ist. Es sind verschiedene Erweiterungen der Webanwendung denkbar. Für künftige Studierendenprojekte könnte die Webanwendung um eine Sprachauswahl ergänzt werden. Die Texte werden durch Provider bereitgestellt. Diese müssten dementsprechend durch Übersetzungen in andere Sprachen ergänzt werden. Des Weiteren müsste eine Schaltfläche für die Sprachauswahl eingebaut werden.
Außerdem könnte die Webanwendung um eine mobile Version erweitert werden. Diese Weiterentwicklung wird durch die bereits bestehende Verwendung einer Single-Page-Anwendung begünstigt. Durch den strukturierten und kommentierten Code, das Drehen eines Erklärvideos und den praxisbezogenen Berichts können sich zukünftige Projektteams schnell in die bestehende Lösung einarbeiten.
Die Projektarbeit hat uns einen Einstieg in die Webentwicklung ermöglicht. Hierbei haben wir das Webframework React erlernt. Des Weiteren konnten wir Kenntnisse über den Entwicklungsprozess von Software-Projekten vertiefen.

A.Anhang

A.1. Zeitplan

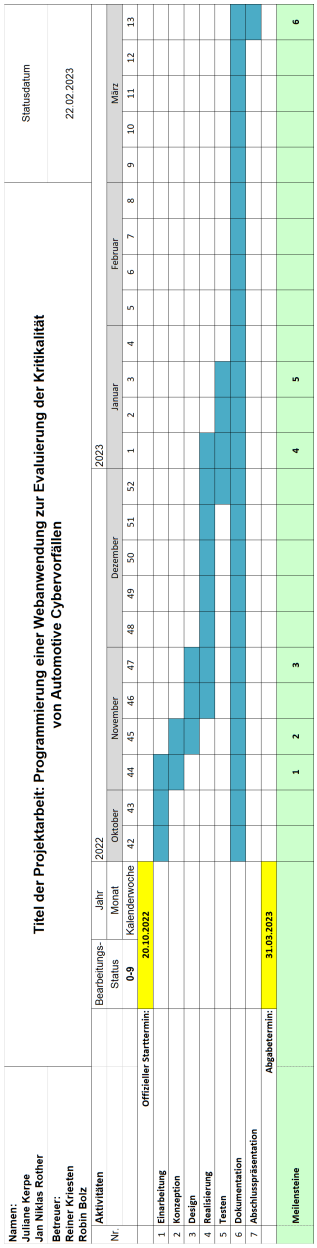

Abbildung A.1.: Zeitplan

A.2. Anforderungen

Req.Nr	Beschreibung	Kategorie	MUSS	SOLL
1	Das Tool muss aus 2 Hauptkomponenten bestehen: i.) Serverseitige math. Berechnung der Karenzzeit ii.) Clientseitige I/O- Komponente zur Interaktion des Benutzers mit i.)	Funktionell	x	
2	Das Tool muss voll webfähig sein. Komponente i.) ist auf einem Webserver gespeichert und wird auch dort ausgeführt.	Funktionell	x	
3	Die Kommunikation (I/O-Datentransfer) muss mindestens mit einem Webbrowser mit einem Desktop-PC mit folgenden Webbrowsern möglich sein: - Google Chrome - Microsoft Edge - Safari (Apple) - Mozilla FireFox	Technisch	x	
4	Für den Webzugang zum Tool ist keine Authentifizierung notwendig	Funktionell	x	
5	Die Kommunikation vom Browser mit dem Webserver muss mit dem HTTPS Protokoll geschützt sein.	Technisch	x	
6	Benutzereingaben und Ausgabedaten für i.) müssen bei Start eines neuen Programmdurchlaufs überschrieben werden.	Technisch	x	
7	Es dürfen keine Eingabedaten gespeichert werden.	Technisch	x	
8	Komponente ii.) muss auf englischer Sprache verfügbar sein.	Funktionell	x	
9	Komponente ii.) soll auf deutscher Sprache verfügbar sein.	Funktionell		x
10	Komponente ii.) muss folgende Komponenten enthalten: 1. Eingabeelemente der Werte RV, S, EM, RL, RC, ED, ISC, SCS, R, APP, CA 2. Eingabeelement des Werts T_0 3. Eingabeelemente des Werts für R und APP (falls von Nutzendem gewünscht) 4. Button zum Ausführen der Berechnung 5. Ausgabeelemente der Variablen TCF, PCF, T_0, GP', T_GP	Funktionell	x	
11	Komponente ii.) soll folgende Komponenten enthalten: 1. Graph zur Visualisierung von GP'	Funktionell		x
12	Das Programm muss falsche bzw. fehlerhafte Eingaben abfangen, sodass es nicht abstürzt.	Technisch	x	
13	Der Nutzende soll zugunsten der Usability über falsche bzw. fehlende Eingaben informiert werden.	Funktionell		x
14	Der Nutzende soll beispielsweise über Tooltips Hilfe zu seiner jeweiligen Eingabe erhalten	Funktionell		x

Abbildung A.2.: Anforderungsliste

ALT	Req.Nr.	Beschreibung	Kategorie	MUSS	SOLL	Test	Erfüllt?	Kommentar
1	1	Das Tool muss dem Benutzer Schaltflächen zur Verfügung stellen, über welche er die Input-Parameter eingeben kann, welche als Berechnungsgrundlage für die Karenzzeit dienen	Funktionell	x		Die Web-Application wird über die URL https://pppster.github.io/gpc mit Google Chrome (Version 110.0.5481.178) aufgerufen. Der Test ist bestanden, wenn alle Input-Parameter und alle Auswahlmöglichkeiten pro Input-Parameter vorhanden sind.	ja	
2	2	Das Tool muss voll webfähig sein.	Funktionell	x		Die Web-Application wird über die URL https://pppster.github.io/gpc mit Google Chrome (Version 110.0.5481.178) aufgerufen. Der Test ist bestanden, wenn die Web-Application nach dem Aufruf bereitgestellt wird	ja	
2	3	Die Kommunikation (I/O-Datentransfer) muss mindestens mit einem Webbrowser mit einem Desktop-PC mit folgenden Webbrowsern möglich sein: - Google Chrome - Microsoft Edge - Safari (Apple) - Mozilla FireFox	Technisch	x		Die Web-Application wird über die URL https://pppster.github.io/gpc mit Google Chrome (Version 110.0.5481.178) aufgerufen und die aufgeführten Tests durchgeführt	nein	Es scheint ein generelles Problem bei Safari und Mozilla Firefox zu sein, dass man in number-Inputs auch Sonderzeichen und Buchstaben eingeben kann. Hier einige Quellen: https://github.com/facebook/react/issues/3810 https://github.com/mui/material-ui/issues/18923
4	4	Für den Webzugang zum Tool ist keine Authentifizierung notwendig	Funktionell	x		Die Web-Application wird über die URL https://pppster.github.io/gpc mit Google Chrome (Version 110.0.5481.178) aufgerufen. Der Test ist bestanden, wenn keine Authentifizierung nötig ist.	ja	
5	5	Die Kommunikation vom Browser mit dem Webserver muss mit dem HTTPS Protokoll geschützt sein.	Technisch	x		Die Web-Application wird über die URL https://pppster.github.io/gpc mit Google Chrome (Version 110.0.5481.178) aufgerufen. Der Test ist bestanden, wenn die Web-Application nach dem Aufruf bereitgestellt wird	ja	
6	6	Benutzereingaben und Ausgabedaten müssen bei Start eines neuen Programmdurchlaufs überschrieben werden.	Technisch	x		Die Web-Application wird über die URL https://pppster.github.io/gpc mit Google Chrome (Version 110.0.5481.178) aufgerufen. Für jeden Input-Parameter wird der default-Wert ausgewählt. Danach wird die Berechnung durchgeführt. Danach wird für jeden Input-Parameter ein Wert ausgewählt, der vom default-Wert abweicht. Der Test ist bestanden, wenn die Benutzereingaben und Ausgabedaten überschrieben werden	ja	
6	7	Es dürfen keine Eingabedaten gespeichert werden.	Technisch	x		Die Web-Application wird über die URL https://pppster.github.io/gpc mit Google Chrome (Version 110.0.5481.178) aufgerufen. Für jeden Input-Parameter wird ein Wert ausgewählt, der vom default-Wert abweicht. Danach wird die Berechnung durchgeführt. Nach der Berechnung wird die Seite neu geladen. Der Test ist bestanden, wenn alle Input-Parameter auf den default-Wert zurückgesetzt sind und die Berechnungsergebnisse zurückgesetzt werden	ja	
N	8	Komponente ii.) muss auf englischer Sprache verfügbar sein.	Funktionell	x		Die Web-Application wird über die URL https://pppster.github.io/gpc mit Google Chrome (Version 110.0.5481.178) aufgerufen. Der Test ist bestanden, wenn die Web-Application in englischer Sprache zur Verfügung steht.	ja	
N	9	Komponente ii.) soll auf deutscher Sprache verfügbar sein.	Funktionell		x	Die Web-Application wird über die URL https://pppster.github.io/gpc mit Google Chrome (Version 110.0.5481.178) aufgerufen. Der Test ist bestanden, wenn die Web-Application in deutscher Sprache zur Verfügung steht.	nein	
N	10	Die Web-Application muss folgende Komponenten enthalten: 1. Eingabeelemente der Werte RV, S, EM, RL, RC, ED, ISC, SCS, R, APP, CA 2. Eingabeelement des Werts T_0 3. Eingabeelemente des Werts für R und APP (falls von Nutzendem gewünscht) 4. Button zum Ausführen der Berechnung 5. Ausgabeelemente der Variablen TCF, PCF, T_0, GP', T_GP	Funktionell	x		Die Web-Application wird über die URL https://pppster.github.io/gpc mit Google Chrome (Version 110.0.5481.178) aufgerufen. Der Test ist bestanden, wenn alle in der Beschreibung zu Req.-Nr. 10 aufgeführt Koponenten enthalten sind.	ja	
N	11	Die Web-Application soll folgende Komponenten enthalten: 1. Graph zur Visualisierung von GP'	Funktionell		x	Die Web-Application wird über die URL https://pppster.github.io/gpc mit Google Chrome (Version 110.0.5481.178) aufgerufen. Der Test ist bestanden, wenn der Graph zur Visualisierung von GP' enthalten ist.	ja	
N	12	Das Programm muss falsche bzw. fehlerhafte Eingaben abfangen, sodass es nicht abstürzt.	Technisch	x		Die Web-Application wird über die URL https://pppster.github.io/gpc mit Google Chrome (Version 110.0.5481.178) aufgerufen. Der Test ist bestanden, wenn folgende fehlerhaften Eingaben abgefangen werden: i) Eingabe der Sonderzeichen: "!", "?" und "=" ii) Eingabe der Buchstaben: "a", und "B"	ja	
N	13	Der Nutzende soll zugunsten der Usability über geänderte Eingaben informiert werden.	Funktionell		x	Die Web-Application wird über die URL https://pppster.github.io/gpc mit Google Chrome (Version 110.0.5481.178) aufgerufen. Der Test ist bestanden, wenn der User nach Änderung eines Input-Parameters auf die Änderung hingewiesen wird.	ja	
N	14	Der Nutzende soll beispielsweise über Tooltips Hilfe zu seiner jeweiligen Eingabe erhalten	Funktionell		x	Die Web-Application wird über die URL https://pppster.github.io/gpc mit Google Chrome (Version 110.0.5481.178) aufgerufen. Der Test ist bestanden, wenn für jede Komponente in der Beschreibung von Req.-Nr. 10 eine Beschreibung in Form eines Hilfstextes vorhanden ist.	ja	
N	15	Die Ergenisse der Grace Period Berechnung müssen mit denen des Excel-Tools übereinstimmen	Funktionell	x		Die Web-Application wird über die URL https://pppster.github.io/gpc mit Google Chrome (Version 110.0.5481.178) aufgerufen. Der Test ist bestanden, wenn die Ergebnisse für die Grace Period Calculation mit denen des Excel-Tools übereinstimmen	teilweise	Da die LUT durch einen Regression angenähert wird, kommt es zu kleinen Abweichungen für die Werte GP und T_GP

Abbildung A.3.: Anforderungsliste mit Tests

A.3. Quellcode

Der Quellcode ist auf github abgelegt und findet sich unter folgendem Link:
`https://github.com/pppster/gpc`

Abbildungsverzeichnis

2.1. Mögliche Antwortmöglichkeiten und Werte der Parameter 3
2.2. Excel-Tool mit (1) User-Eingaben, (2) Berechnung-Button, (3) Ergebnissen und (4) Diagramm 4
2.3. Auszug aus LUT des Excel-Tools 5

4.1. UML Kompositionsstrukturdiagramm 7
4.2. UML Klassendiagramm 9
4.3. Mockup 1 11
4.4. Mockup 2 12
4.5. Mockup 3 13
4.6. Entscheidungsmatrix zum Webframework 14

5.1. Vollbild des fertigen Webframeworks 16
5.2. oben: Textfeld zu APP verborgen / unten: Textfeld zu APP angezeigt 17
5.3. Beispielhafter Tooltip 18
5.4. Instanz der Klasse *ValueInput* 19
5.5. Instanz der Klasse *TextInput* 20
5.6. Instanz der Klasse *DisplayValues* 20
5.7. Instanz der Klasse *Plot* 21
5.8. Instanz der Klasse *ToolTip* 21
5.9. Anzunäherndes Diagramm aus dem Excel-Tool 22
5.10. Approximierte Normalverteilung mit verändertem Scope (links) und unverändertem Scope (rechts) 23
5.11. Stückweise Regression 24
5.12. UML Sequenzdiagramm 25
5.13. Ordnerstruktur 28

6.1. Homepage-Property 33
6.2. deployment 34

7.1. Automatisiert extrahierte Berechnungsergebnisse 36

A.1. Zeitplan 40
A.2. Anforderungsliste 41

A.3. Anforderungsliste mit Tests 42

Abkürzungsverzeichnis

APP	Certification and Approval
CA	Contractual Agreements
CDF	Cumulative Distribution Function
CSV	Comma Separated Values
ED	Exploit Code Dissemination
EM	Exploit Code Maturity
IEEM	Institut für Energieeffiziente Mobilität
ISC	Incident Scale
JSON	JavaScript Object Notation
LUT	Look-Up Table
MPA	Multi Page Application
PCF	Process Criticality Factor
R	Remediation Dissemination
RC	Report Confidence
RL	Remediation Level
RV	Vulnerability Risk Value
SCS	Supply Chain Scale
S	Scope
SPA	Single Page Application
TCF	Time Criticality Factor
UML	Unified Modeling Language
VBA	Visual Basic For Application

Literaturnachweise

[1] *Global Automotive Cybersecurity Report 2023*. https://upstream.auto/2023report/, . – Accessed on 17.03.2023

[2] *The Automotive Security Research Group Disclosure Policy*. https://asrg.io/disclosure/, . – Accessed on 17.03.2023

[3] BOLZ, Robin: Evaluating reasonable patching times for security product vulnerabilities in the automotive field. Version: 2023. http://dx.doi.org/10.5281/zenodo.7573669. In: FESSLER, Dirk (Hrsg.) ; KETTNER, Maurice (Hrsg.) ; KRIESTEN, Reiner (Hrsg.) ; NENNINGER, Philipp (Hrsg.) ; OFFERMANN, Peter (Hrsg.): *Reports on Energy Efficient Mobility*. Karlsruhe : IEEM, 2023. – DOI 10.5281/zenodo.7573669, S. 140–156

[4] BRANDT-POOK, Hans ; KOLLMEIER, Rainer: *Softwareentwicklung kompakt und verständlich*. Springer, 2020. http://dx.doi.org/10.1007/978-3-658-30631-1. http://dx.doi.org/10.1007/978-3-658-30631-1. – ISBN 978–3–658–30631–1

[5] SMITH, J.: *Web Development with JavaScript and AJAX Illuminated*. 1. Burlington, MA : Jones & Bartlett Learning, 2018

[6] DEVORE, Jay L.: *Probability and Statistics for Engineering and the Sciences*. 9th. Cengage Learning, 2016