



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law

## Software-Engineering I

---

# Algosim

---

vorgelegt am 30. Oktober 2023

**Jahrgang:** 2022 B

**Gruppe:** 4

**Mitglieder:** Bartels, Paul Anton  
Berner, Philip  
Karacam, Onur  
Krause, Nick  
Waldmann, Marvin

# Inhaltsverzeichnis

Abbildungsverzeichnis	II
Tabellenverzeichnis	II
1 Kurzbeschreibung Produkt	1
2 Projektsteckbrief	2
3 Ziele	3
4 Projektteam und Rollen	6
5 Risikomanagement	9
6 Aufwände	11
7 Produktstruktur	15
8 Herausforderungen und Lösungen	16
9 UML: Anwendungsfalldiagramm	18
9.1 Vollständiger Überblick . . . . .	18
9.2 Detaildiagramm . . . . .	24
10 UML: Aktivitätsdiagramm	28
11 Qualitätskriterien	31
12 Technologien und Produkte	32
13 Verwendungsanleitung Lösung	34
13.1 Webseite . . . . .	34
13.2 Quellcode . . . . .	34
13.3 Docker-Compose . . . . .	35
Anhang	37
A Risikoregister	38
B Qualitätsregister	39

# Abbildungsverzeichnis

1	Risikoprofil / Risikomatrix . . . . .	10
2	Produktstrukturplan . . . . .	15
3	Use-Case-Diagramm: Überblick . . . . .	19
4	Use-Case-Diagramm: Detailansicht . . . . .	27
5	Aktivitätsdiagramm . . . . .	29
6	BFS ausführen – Teil des Aktivitätsdiagramms . . . . .	30

# Tabellenverzeichnis

1	Projektsteckbrief . . . . .	2
2	Projektzielübersicht . . . . .	5
3	AKV-Matix des Projektmanagers . . . . .	6
4	AKV-Matix des Entwicklers . . . . .	6
5	AKV-Matix des Code-Reviewers . . . . .	7
6	AKV-Matix des Testers . . . . .	7
7	AKV-Matix des DevOps-Ingenieurs . . . . .	7
8	AKV-Matix des Grafikdesigners . . . . .	7
9	AKV-Matix des Paperautors . . . . .	8
10	Rollenverteilung . . . . .	8
11	Aufwandsschätzung . . . . .	12
12	Tatsächlicher Aufwand und Abweichung von der Schätzung . . . . .	13
13	Textuelle Beschreibung: Quiz spielen . . . . .	18
14	Textuelle Beschreibung: Quizseite aufrufen . . . . .	19
15	Textuelle Beschreibung: Startseite aufrufen . . . . .	20
16	Textuelle Beschreibung: Sortierseite aufrufen . . . . .	20
17	Textuelle Beschreibung: Suchseite aufrufen . . . . .	21
18	Textuelle Beschreibung: Algorithmus wählen . . . . .	21
19	Textuelle Beschreibung: Beschreibung anzeigen . . . . .	22
20	Textuelle Beschreibung: Algorithmus ausführen . . . . .	23
21	Textuelle Beschreibung: Antwort prüfen . . . . .	24

22	Textuelle Beschreibung: Korrekte Lösung anzeigen . . . . .	24
23	Textuelle Beschreibung: Erneut versuchen . . . . .	25
24	Textuelle Beschreibung: Neue Frage generieren . . . . .	25
25	Textuelle Beschreibung: Antwort auswählen . . . . .	26
26	Textuelle Beschreibung: Algorithmus ausführen . . . . .	26

# 1 Kurzbeschreibung Produkt

Wer Mathematik oder Informatik studiert beziehungsweise sich anderweitig mit deren Inhalten auseinandersetzt, wird früh auf die sogenannten Such- und Sortieralgorithmen treffen. Beim ersten Kontakt und mit unzureichendem Unterrichtsmaterial können diese komplexen Themen unverständlich wirken. Deshalb wollen wir durch Visualisierungen beim Lernen und Verstehen der Algorithmen helfen.

Wir sind eine Gruppe aus fünf dualen Studenten der HWR Berlin und haben uns im Rahmen des bisherigen Studiums mit den oben genannten Themen beschäftigt. Aus unseren eigenen anfänglichen Schwierigkeiten nehmen wir die Motivation, um besagte Themen anderen besser zu vermitteln.

Das Ziel für dieses Projekt ist die Erstellung einer zentralen Webseite, welche Sortier- und Suchalgorithmen visualisieren kann. Dabei soll der Ausgangszustand, auf dem die Algorithmen arbeiten, durch den Nutzer bestimmt werden. Sortieralgorithmen sollen durch unterschiedlich große Säulen visualisiert werden, besondere Positionen in der Liste werden farbig hervorgehoben. Für die Darstellung der Suchalgorithmen wird ein Gitter aus Wänden und Wegen genutzt, welches sich in einen Graphen mit Knoten und Kanten überführen lässt. Durch die farbliche Markierung von besuchten Knoten und der Darstellung von Entfernungen im Gitter kann ein Suchalgorithmus visualisiert werden. Zu jedem Algorithmus wird außerdem eine umfangreiche Erläuterung formuliert, welche für den Nutzer einsehbar ist.

Da die Zahl der Such- und Sortieralgorithmen schier grenzenlos ist, haben wir uns entschlossen, nur eine Auswahl der relevantesten Algorithmen zu implementieren. Mit unserer Auswahl soll ein Überblick über die vergleichsbasierten Sortieralgorithmen und die graphentheoretischen Suchalgorithmen gegeben werden. Somit kann ein Nutzer die Vorgehensweise sowohl logisch als auch visuell verstehen.

Zum selbstständigen Vertiefen der Lerninhalte wird ein Quiz bereitgestellt, welches einen Algorithmus visualisiert und den Nutzer fragt, um welchen es sich handelt. Wir versprechen uns vom Quiz einen erhöhten Lernerfolg, da es sich um eine interaktive und unterhaltsame Wissensabfrage handelt.

## 2 Projektsteckbrief

Gruppe:	4	Projektname:	Algosim
Projektauftraggeber:	Hr. Kretzmer	Projektleiter:	Hr. Berner
Kurzbeschreibung des Projekts:	Visualisierung von Sortieralgorithmen und Suchalgorithmen inklusive eines Quiz und Algorithmusbeschreibungen. Das Produkt ist eine Webseite.		
Projektstart:	Wahl des Projektthemas		21.08.2023
Projektende:	Abgabe bei Moodle		30.10.2023
Projektdauer:	70 Tage		
Projektziele:	<p>Pünktliche Abgabe</p> <p>Darstellung von Sortieralgorithmen</p> <p>Darstellung von Suchalgorithmen</p> <p>Bereitstellung von beschreibenden Texten</p> <p>Implementation eines Quiz</p>		
Meilensteine:	<p>M0: Projektstart</p> <p>M1: Fertigstellung Sortieralgorithmen</p> <p>M2: Fertigstellung Suchalgorithmen</p> <p>M3: Einbinden der Quiz-Seiten</p> <p>M4: Fertigstellen der Landingpage mit Logo</p> <p>M5: Projektabschluss</p>		
Projektressourcen und Projektbudget:	Personalbudget:	320 Ph	
	Deploymentbudget:	10 EUR	
Projektmitglieder:	<p>Bartels, Paul Anton (Paperautor, ...)</p> <p>Berner, Philip (Projektmanager, ...)</p> <p>Karacam, Onur (Entwickler, ...)</p> <p>Krause, Nick (Grafikdesigner, ...)</p> <p>Waldmann, Marvin (DevOps-Ingenieur, ...)</p>		
Hauptrisiken:	Technologien des Front-Ends sind nicht leistungsfähig genug für Berechnungen; langfristiger Ausfall von Mitgliedern		

Tabelle 1: Projektsteckbrief

### 3 Ziele

Damit der Erfolg des Projekts in der Retrospektive ermittelt werden kann, ist es nötig, Projektziele zu definieren. Im Rahmen einer Diskussion konnten sich die Gruppenmitglieder auf die folgende Liste an Zielen einigen.

Nr.	Klassifizierung	Beschreibung	Messkriterium	Priorität
1	Hauptziel	Darstellung der Sortieralgorithmen	Sortieralgorithmen werden durch ein Balkendiagramm dargestellt und können schrittweise abgespielt werden. Vergleichen und Vertauschen werden als einzelne Schritte dargestellt.	
2	Leistung	Implementierte Sortieralgorithmen	BubbleSort, InsertionSort, QuickSort, MergeSort sind implementiert.	Muss-Ziel
3	Leistung	Benutzerdefinierte Ausgangsmenge	Zu sortierende Werte können durch den Benutzer bestimmt werden.	Soll-Ziel
4	Hauptziel	Darstellung der Suchalgorithmen	Suchalgorithmen, die einen Weg zwischen dem Start- und Endknoten eines Graphen finden, werden dargestellt und können schrittweise abgespielt werden.	
5	Leistung	Implementierte Suchalgorithmen	Breitensuche, Tiefensuche, Dijkstra, A* sind implementiert.	Muss-Ziel
6	Leistung	Benutzerdefinierte Ausgangsgraphen	Zu durchsuchende Graphen können durch den Benutzer gestaltet werden.	Soll-Ziel

### 3 Ziele

7	Leistung	Vordefinierte Ausgangsgraphen	Es stehen vordefinierte Graphen zur Verfügung.	Kann-Ziel
8	Leistung	Beschreibungen	Jeder Algorithmus besitzt einen beschreibenden Text.	Muss-Ziel
9	Leistung	Quizseite	Dem Benutzer wird ein zufälliger Algorithmus gezeigt. Er kann den Algorithmus auswählen, den er glaubt, erkannt zu haben und erfährt danach, ob seine Lösung korrekt ist und kann gegebenenfalls einen neuen Algorithmus anschauen, eine neue Antwort geben oder die richtige Lösung erhalten.	Soll-Ziel
10	Leistung	Startseite	Eine Startseite erklärt das Projekt und die Hintergründe.	Soll-Ziel
11	Leistung	Logo	Ein Logo ist vorhanden.	Soll-Ziel
12	Termin	Abschluss der Entwicklung	Die Entwicklung der Anwendung wird am 23.10.2023 bis 19:00 Uhr abgeschlossen.	Soll-Ziel
13	Termin	Präsentation	Es wird eine Abschlusspräsentation inklusive Slideshow nach den Prüfungsanforderungen am 30.10.2023 gehalten.	Muss-Ziel
14	Termin	Projektabschluss	Upload der Abschlusspräsentation, Software und des Papers bis zum 30.10.2023 um 23:59 Uhr.	Muss-Ziel



### 3 Ziele

---

15	Sozial	Arbeitslast	Arbeitsaufwand von maximal 20 h pro Woche je Teammitglied.	Soll-Ziel
16	Budget	Gesamtzeitaufwand	Der Projektaufwand beträgt maximal 40 Personentage.	Soll-Ziel
17	Kosten	Produktivsetzung	Die Produktivsetzung kostet maximal 10 EUR.	Soll-Ziel
18	Nicht-Ziel	Nicht-Vergleichsbasierte Sortieralgorithmen	Nicht-Vergleichsbasierte Sortieralgorithmen werden nicht dargestellt.	

Tabelle 2: Projektzielübersicht

## 4 Projektteam und Rollen

Bei der Bearbeitung eines Projekts ist es wichtig, den verschiedenen Teammitgliedern jeweils eigene Aufgabenbereiche und Verantwortlichkeiten zuzuschreiben, damit die Zusammenarbeit reibungslos ablaufen kann. Durch das Verteilen der Aufgaben können die Kompetenzen und Erfahrungen der Gruppenmitglieder optimal genutzt werden. Außerdem wird verhindert, dass unnötiger Mehraufwand entsteht, wenn zu viele Gruppenmitglieder dieselbe Rolle ausüben. Deshalb haben wir für die Erarbeitung von Algosim verschiedene Rollen definiert, welche in den folgenden AKV-Matrizen erläutert werden.

Aufgaben	Arbeitspakete bzw. Tickets angemessener Größe definieren; Einhaltung von internen und externen Fristen kontrollieren; Pflege der Ist-Aufwände und Qualitätskriterien überwachen; Projektmitgliedern bei Problemen zur Seite stehen
Kompetenzen	Tickets erstellen; bei Nichteinhaltung von Fristen Projektmitglieder ermahnen; Projektmitglieder auf die Ist-Aufwände bzw. Qualitätskriterien hinweisen
Verantwortlichkeiten	Projekterfolg und fristgerechte Projektabgabe sicherstellen; regelmäßige Statusüberwachung

Tabelle 3: AKV-Matrix des Projektmanagers

Aufgaben	Umsetzung von Tickets in einem eigenen Branch; nach der Bearbeitung eines Tickets eine Pull-Request eröffnen; Entwicklertests durchführen; Unit-Tests implementieren; Texte, welche innerhalb des Produktes dargestellt werden, schreiben
Kompetenzen	Feature-Branches erstellen; Pull-Requests eröffnen; Bearbeitungsstatus von Tickets setzen; Tickets sich selbst zuweisen
Verantwortlichkeiten	funktionsfähiger und stabiler Code; sinnvolle Unit-Tests implementieren; Tickets möglichst fristgerecht umsetzen

Tabelle 4: AKV-Matrix des Entwicklers

Aufgaben	Pull-Requests bearbeiten; Quellcode von Entwicklern auf Fehler und Unsauberkeiten kontrollieren; Verbesserungsempfehlungen geben
Kompetenzen	Pull-Requests akzeptieren und mergen (Entwicklungs-Branch und Haupt-Branch); Change-Requests erstellen; Tickets von akzeptierten Pull-Requests schließen
Verantwortlichkeiten	Codequalität sicherstellen; zeitnahe Bearbeitung von Pull-Requests

Tabelle 5: AKV-Matix des Code-Reviewers

Aufgaben	Funktions- und Integrationstests durchführen; Einhaltung der Qualitätskriterien prüfen und dokumentieren; Projektmanager auf gefundene Fehler bzw. nicht eingehaltene Qualitätskriterien hinweisen
Kompetenzen	Status von Qualitätskriterien aktualisieren
Verantwortlichkeiten	Aktualität der Tabelle zu den Qualitätskriterien sicherstellen; regelmäßige Prüfung von Funktionen

Tabelle 6: AKV-Matix des Testers

Aufgaben	Einrichtung der GitHub-Repositories; Grundstruktur vom Projekt und des Papers erstellen; Rechte der Gruppenmitglieder innerhalb der GitHub-Repositories verwalten; CI- und CD-Pipelines definieren, sofern diese benötigt werden
Kompetenzen	GitHub-Repository erstellen; Rechte von Mitgliedern innerhalb eines Repositories verändern; CI/CD-Pipelines definieren
Verantwortlichkeiten	funktionstüchtige Entwicklungsumgebung sicherstellen; Rechte innerhalb des Repositories nur bei Bedarf vergeben

Tabelle 7: AKV-Matix des DevOps-Ingenieurs

Aufgaben	Logo für das Projekt entwerfen; Grafiken erstellen; Layout der Seite ansprechend gestalten
Kompetenzen	Pull-Requests erstellen; Tickets für benötigte Grafiken bearbeiten
Verantwortlichkeiten	sicherstellen, dass das Produkt ansprechend aussieht; Logo fristgerecht erstellen

Tabelle 8: AKV-Matix des Grafikdesigners

Aufgaben	Kapitel für das Paper schreiben; Kapitel von anderen Paperautoren Korrektur lesen
Kompetenzen	Pull-Requests im Paper-Repository erstellen; Pull-Requests anderer Autoren im Paper-Repository bearbeiten
Verantwortlichkeiten	Paper fristgerecht fertigstellen; Richtigkeit des Paperinhalts sicherstellen

Tabelle 9: AKV-Matrix des Paperautors

Das Projektteam besteht aus den 5 Mitgliedern Marvin Waldmann, Onur Karacam, Philip Berner, Paul Anton Bartels und Nick Krause. Die Rollenverteilung innerhalb des Projekts geschieht nach den Kompetenzen, Erfahrungen und Interessen der Projektmitglieder. Die Rolle des Code-Reviewers setzt Wissen im Umgang mit der genutzten Programmiersprache voraus, weshalb beim Verteilen dieser Rolle besonders auf solche Erfahrungen geachtet wurde. Auch der DevOps-Ingenieur benötigt spezielles Vorwissen, weshalb nur Gruppenmitglieder mit Erfahrung im Umgang mit Git-Repositories diese Rolle einnehmen. Es folgt eine Tabelle, welche die Gruppenmitglieder und die von ihnen ausgeführten Rollen auflistet.

Teammitglied	Projektmanager	Entwickler	Code-Reviewer	Tester	DevOps-Ingenieur	Grafikdesigner	Paperautor
Paul Anton Bartels		x	x	x	x		x
Onur Karacam		x		x			
Nick Krause		x		x		x	
Marvin Waldmann	x	x	x	x	x		
Philip Berner	x	x	x	x			x

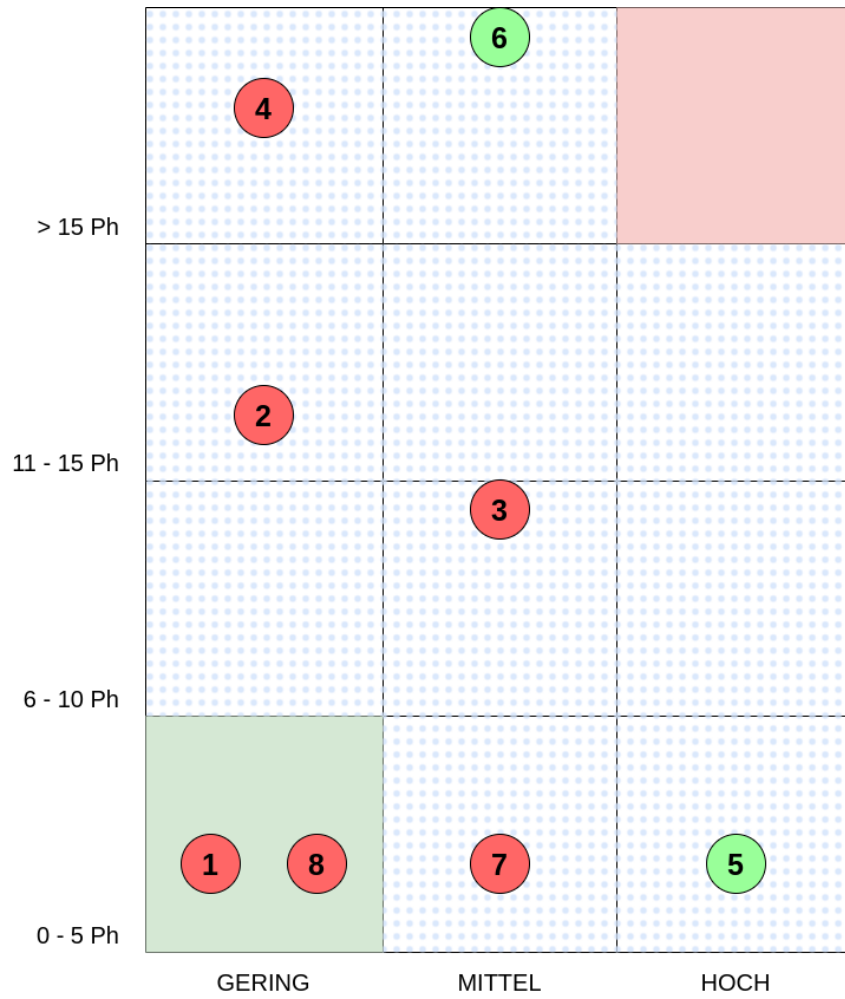
Tabelle 10: Rollenverteilung

## 5 Risikomanagement

In der Realität können jederzeit Ereignisse eintreten, die den geplanten Ablauf eines Projekts zu verändern drohen. Diese werden Risiken genannt und in Bedrohungen und Chancen unterteilt, je nachdem, ob sie das Projekt negativ oder positiv beeinflussen.

Für einen zuverlässigen Ablauf des Projekts ist es wichtig, alle Ereignisse zu identifizieren und zu bewerten. Dafür wurde im Anhang A auf der Seite 38 ein auszugsweises Risikoregister erarbeitet. In diesem werden die potenziellen Risiken beschrieben und nummeriert sowie in ihrer Eintrittswahrscheinlichkeit und Auswirkung bewertet. Anschließend werden sie nach ihrer Art und dem Typ (Bedrohung oder Chance) kategorisiert. Um die Fertigstellung des Projekts zu sichern, wird abschließend für jedes Risiko eine Behandlungsmethode bestimmt und beschrieben.

Zur übersichtlicheren Darstellung der Risiken wird außerdem ein Risikoprofil erstellt. Dieses zeigt alle Risiken auf einer Matrix zwischen Eintrittswahrscheinlichkeit und Auswirkung. Unser folgendes Risikoprofil nutzt als Skala für die Eintrittswahrscheinlichkeit die Einschätzungen gering, mittel und hoch. Die Auswirkung wird in 0–5, 6–10, 11–15 und >15 Personenstunden eingeteilt.



Legende:

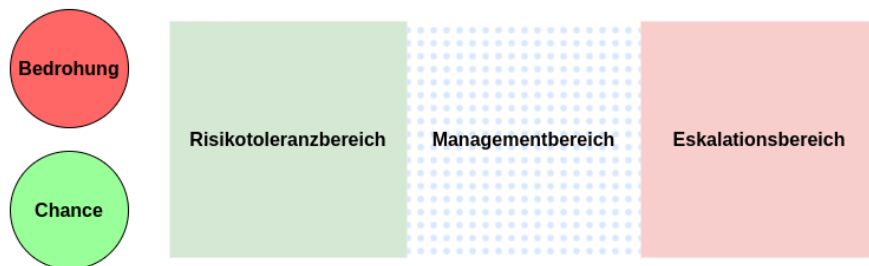


Abbildung 1: Risikoprofil / Risikomatrix

## 6 Aufwände

Im Rahmen der Vorbereitung des Projekts wird eine Aufwandsschätzung durchgeführt. Diese kann genutzt werden, um den Fortschritt des Projekts zu beurteilen und in der Retrospektive die Genauigkeit der Schätzung zu bewerten. Die Schätzung des Aufwands wurde mithilfe der Expertenmethode ermittelt, nach welcher die zuständigen Experten jeweils ihren eigenen Arbeitsaufwand einschätzen.

Der Aufwand des Projekts setzt sich unter anderem aus den Funktionen des Programms und den Inhalten des Papers zusammen. Sowohl Unit-Tests als auch mögliche Fehlerbehebungen sind im Rahmen des Aufwands für die Funktion zu berücksichtigen. Zusätzlich entstehen Aufwände durch Integrationstests, Qualitätssicherung und Code Reviews im Rahmen von Merge-Requests.

Es wird mit einem geringen Besprechungsaufwand gerechnet, da viele Besprechungen nicht mit allen Mitgliedern erfolgen müssen. Weil es sich um ein kleines Team handelt, bleibt der Aufwand für das Projektmanagement überschaubar. Neben dem tatsächlichen Aufwand des Projekts wird auch ein Risiko-Prozentsatz eingerechnet.

Die konkrete Aufwandsschätzung folgt in tabellarischer Form:

Nr.	Aktivitäten	Aufwand in PT	
<b>1</b>	<b>Funktionalitäten</b>		<b>13,50</b>
1.1	Algorithmen		4,00
1.1.1	Sortialgorithmen implementieren	2,00	
1.1.2	Suchalgorithmen implementieren	2,00	
1.2	Visualisierung		5,00
1.2.1	Säulendarstellung für Sortialgorithmen	2,00	
1.2.2	Gitterdarstellung für Suchalgorithmen	3,00	
1.3	Benutzeroberfläche		4,50
1.3.1	Layout	0,50	
1.3.2	Beschreibungen für Sortialgorithmen	1,00	
1.3.3	Beschreibungen für Suchalgorithmen	1,00	
1.3.4	Quiz	2,00	
<b>2</b>	<b>Paper</b>		<b>5,25</b>
2.1	Layout	0,25	
2.2	Kurzbeschreibung Produkt	0,25	
2.3	Projektsteckbrief	0,25	

2.4	Ziele	0,25	
2.5	Projektteam & Rollen	0,25	
2.6	Risikomanagement	0,50	
2.7	Aufwände	0,50	
2.8	Produktstruktur	0,25	
2.9	Herausforderungen & Lösungen	0,50	
2.10	UML: Anwendungsfalldiagramm	0,50	
2.11	UML: Aktivitätsdiagramm	0,50	
2.12	Qualitätskriterien	0,50	
2.13	Technologien & Produkte	0,50	
2.14	Verwendungsanleitung Lösung	0,25	
<b>3</b>	<b>Tests &amp; Qualitätssicherung</b>		<b>5,40</b>
3.1	System- und Integrationstests	0,40	
3.2	Qualitätssicherung (z. B. Code Reviews)	5,00	
<b>4</b>	<b>Dokumentation</b>		<b>0,00</b>
<b>5</b>	<b>Produktivsetzung</b>		<b>0,00</b>
<b>6</b>	<b>Besprechungen</b>		<b>3,13</b>
6.1	Interne Besprechung (5 MA je 0,50h/Woche)	3,13	
<b>7</b>	<b>Projektmanagement</b>		<b>2,50</b>
7.1	Zeitaufwand des Projektleiters (2h/Woche)	2,50	
	<b>Summe Gesamtaufwand</b>		<b>29,78</b>
<b>8</b>	<b>Risiko</b>		<b>3</b>
8.1	Risiko (10 % des Gesamtaufwands)	3	
<b>Summe Gesamtaufwand Projekt</b>			<b>32,78</b>

Tabelle 11: Aufwandsschätzung

Das Team hat im Zuge der Projektbearbeitung die tatsächlich benötigte Zeit gemessen, damit die geschätzten Aufwände den tatsächlichen gegenübergestellt werden können. Die Zusammenfassung erfolgt ebenfalls tabellarisch.



Nr.	Aktivitäten	Aufwand in PT	Abweichung	
			Absolut	Relativ
1	Funktionalitäten	17,27	+3,77	+27,9 %
2	Paper	7,94	+2,69	+51,2 %
3	Tests & Qualitätssicherung	3,70	−1,70	−31,5 %
4	Dokumentation	0,00	±0,00	±00,0 %
5	Produktivsetzung	0,16	+0,16	+∞ %
6	Besprechungen	1,43	−1,70	−54,3 %
7	Projektmanagement	2,03	−0,47	−18,8 %
<b>Summe Gesamtaufwand</b>		<b>32,53</b>	<b>+2,75</b>	<b>+9,2 %</b>
(exkl. Risiko)			<b>−0,25</b>	<b>−0,01 %</b>
		(inkl. Risiko)		

Tabelle 12: Tatsächlicher Aufwand und Abweichung von der Schätzung

In der Tabelle ist erkennbar, dass die Entwicklung von Algosim (Funktionalitäten) und das Schreiben des Papers deutlich mehr Zeit beanspruchten als ursprünglich vermutet. Dies lässt sich — neben optimistischen Schätzungen durch die mangelnde Planungserfahrung des Teams — größtenteils auf nicht antizipierte technische Herausforderungen zurückführen. Dazu gehören unter anderem die Gestaltung größerer Tabellen in  $\text{\LaTeX}$  oder die in Kapitel 8 näher beschriebenen Schwierigkeiten mit der Graphendarstellung in Algosim. Ein weiterer Grund für den zusätzlichen Aufwand beim Paper ist die in der Planung unberücksichtigte Revision.

Im Gegensatz dazu lag die veranschlagte Zeit für Tests und Qualitätssicherung über der tatsächlich benötigten. Die Dokumentation wurde wie geplant nicht durchgeführt und verbrauchte deshalb keinen (zusätzlichen) Arbeitsaufwand. Anders fiel die Entscheidung bei der Produktivsetzung aus. Hier beschloss das Team im Laufe des Projekts, Algosim doch auf einer Webseite bereitzustellen, was einen geringen Mehraufwand mit sich brachte.

Wie bereits bei der Zeitplanung vermutet, blieb der Aufwand für Besprechungen und Projektmanagement gering. Die realen Zahlen lagen hier jedoch deutlich — teilweise um mehr als die Hälfte — unter den Planzahlen. Die Ursache dafür lag in einem guten Teamklima und einer konstruktiven Zusammenarbeit. So konnten viele Themen schnell geklärt und Besprechungen auf ein Minimum reduziert werden.

Letztlich zeigte sich, dass die Zeitgewinne an Besprechungen, Projektmanagement und Qualitätssicherung den Mehraufwand in der Entwicklung und Produktivsetzung teilweise kompensieren konnten. Der Gesamtaufwand liegt zwar weiterhin über dem

geplanten Zeitbudget, überschreitet jedoch nicht den Rahmen des einkalkulierten Risikos.

# 7 Produktstruktur

Vor dem Beginn eines Projekts ist es wichtig, die Inhalte des Endproduktes klar zu definieren. Dabei hilft ein Produktstrukturplan, der die einzelnen Bestandteile rekursiv aufzeigt. Aufbauend auf diesem Plan kann später auch die Zeitplanung durchgeführt werden, indem für jedes Teilprodukt der benötigte Aufwand geschätzt und aufsummiert wird. Des Weiteren können mit der Zerlegung des Produktes auch die Qualitätskriterien der Teilprodukte und des Gesamtwerkes ermittelt werden. In einer gemeinsamen Besprechung in der Projektgruppe entstand dazu der folgende Produktstrukturplan.

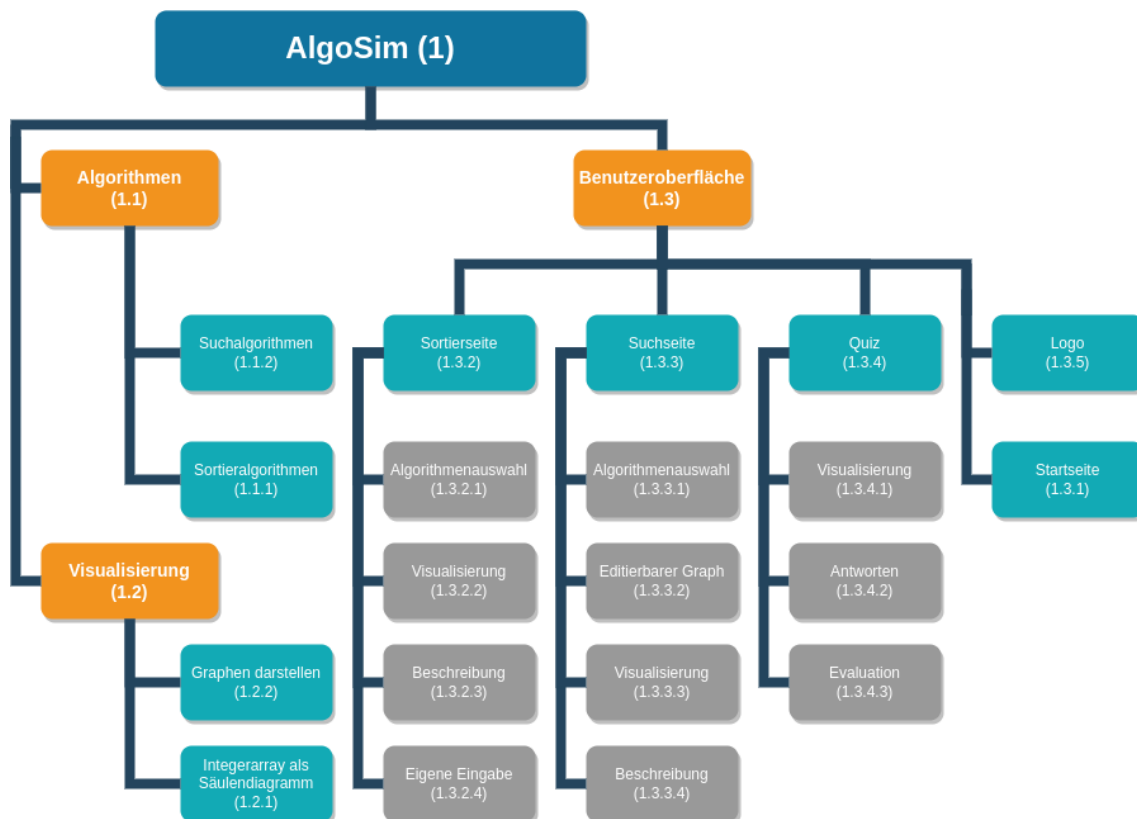


Abbildung 2: Produktstrukturplan

## 8 Herausforderungen und Lösungen

Eine der ersten Herausforderungen, mit denen sich das Projektteam auseinandersetzen musste, war die Festlegung des Produktaufbaus. Aufgrund unterschiedlicher Erfahrungen im Bereich der Softwareentwicklung und abweichenden Präferenzen für Technologien war eine Einigung schwer. Das Team einigte sich jedoch schnell darauf, dass Algosim eine Webanwendung werden sollte. Dies hat den Vorteil, dass die Anwendung auf vielen Geräten abrufbar ist, ohne eine Installation zu benötigen. Weitaus stärker wurde diskutiert, ob Algosim ein Back-End benötigt. Mit einem Back-End würde die Webseite dem Endnutzer weniger Computer-Ressourcen abverlangen und die Simulationen könnten schneller berechnet werden. Es gibt jedoch auch Kritikpunkte. Unter anderem entsteht durch ein Back-End Mehraufwand, da die Kommunikation zwischen beiden Komponenten implementiert werden muss. Außerdem wird auch das Deployment erschwert und die Webseite ist nur in Verbindung mit dem Back-End nutzbar. Aufgrund dieser Argumente hat sich das Team gegen ein Back-End entschieden. Die Wahl des Frameworks für das Front-End stellte sich nach den vorhergehenden Entscheidungen als relativ leicht heraus.

Die Visualisierung von Algorithmen ist die Kernaufgabe von Algosim, die Wahl der Visualisierung präsentierte sich in einigen Fällen jedoch als Herausforderung. Unter anderem passierte dies mit der Darstellung des HeapSort-Algorithmus. Anders als bei vielen anderen Sortieralgorithmen wird die Idee des Algorithmus beim bloßen Hervorheben von Vergleich- und Tauschoperationen nicht sofort klar. Das liegt vor allem an dem Fakt, dass HeapSort auf einem Heap, einer baumartigen Datenstruktur basiert. Damit der Heap in der Darstellung erkannt wird, hat sich der Entwickler dazu entschieden, jede Ebene des Heaps mit einer eigenen Farbe hervorzuheben. Da das zuvor genutzte System nur eine begrenzte Auswahl an Hervorhebungsfarben erlaubte, musste es in Teilen überarbeitet werden, was einen Mehraufwand darstellte.

Auch die Visualisierung der Graphen erwies sich als eine große Herausforderung. Anders als eine Liste aus Zahlen, welche sich leicht als Säulen- oder Balkendiagramm darstellen lässt, ist die Darstellung von Graphen sehr komplex. Dies hängt zum Teil mit den grundlegenden Eigenschaften von Graphen zusammen, denn nicht alle Graphen sind planar. Das bedeutet, dass sich einige Graphen nicht ohne kreuzende Kanten darstellen lassen. Da die Visualisierung eines nicht planaren Graphen eher zu Verwirrung führt, haben sich die Gruppenmitglieder dazu entschieden, solche Graphen zu unterbinden. Dies wird durch eine Darstellung der Graphen

als Gitter erreicht. So kann ein Knoten maximal 4 Kanten besitzen. Wenn sich zwei Kanten kreuzen, entsteht automatisch ein Knoten und der Graph bleibt planar.

Außerdem kam es zu Schwierigkeiten bei der Darstellung von informierten Suchalgorithmen (Dijkstra und A\*). Diese Suchalgorithmen schreiben den Knoten verschiedene Werte zu, welche für das Verständnis des Nutzers von Bedeutung sind. Die Entwickler haben sich darauf geeinigt, dass es sinnvoll ist, die Werte auf den Knoten als Text darzustellen. Sollten mehrere Werte vorhanden sein, wird die Bedeutung durch einen Buchstaben spezifiziert. In der Beschreibung des Algorithmus werden die Werte und ihre Beschriftung genauer erläutert, damit die Visualisierung eindeutig bleibt. Zudem wurde entschieden, dass die initialen Werte, welche häufig  $\infty$  sind, nicht dargestellt werden, damit der Fokus auf die relevanten Werte gerichtet wird.

Eine weitere Herausforderung, welche sich im Zuge der Entwicklung präsentierte, war das Generieren von Quizfragen für das Suchalgorithmenquiz. Hierbei war nicht die Wahl des Algorithmus selbst das Problem, sondern das Generieren eines validen Graphen, welcher den Algorithmus eindeutig erkennbar macht. Das Team erkannte, dass das zufällige Generieren einen erheblichen Mehraufwand verursachte und nicht zwingend die Benutzererfahrung verbesserte. Deshalb entschied man sich dazu, stattdessen einen Graphen aus einer Liste von vorgefertigten Graphen zu wählen, welche in einer Art und Weise konstruiert sind, dass der Algorithmus eindeutig erkennbar ist.

## 9 UML: Anwendungsfalldiagramm

Anwendungsfalldiagramme zeigen, wie verschiedene Akteure mit einem System interagieren. Dabei kann es sich beispielsweise um Nutzer oder Systemadministratoren handeln, die mit einem neuen Programm arbeiten werden. Ziel dieser Analyse ist es, die Anwendungsmöglichkeiten einer Software zu definieren und Abhängigkeiten zwischen diesen Use-Cases erkennbar zu machen.

Das Algosim-Projektteam hat ein grobes Use-Case-Diagramm erarbeitet, dass alle wichtigen Anwendungsfälle des gesamten Programmes darstellt. Außerdem wurde für den Use-Case *Quiz spielen* ein detaillierteres Diagramm erstellt. Jeder Anwendungsfall erhält zudem eine textuelle Beschreibung. Diese schildert Details, welche in einer Grafik nicht dargestellt werden können.

### 9.1 Vollständiger Überblick

Name	Quiz spielen
Kurzbeschreibung	Der Nutzer bekommt verschiedene zufällige Listen oder Graphen, die von einem unbekannten Algorithmus sortiert oder durchsucht werden. Er muss dann erkennen, welcher Algorithmus gezeigt wurde. Für eine ausführlichere Beschreibung siehe Kapitel 9.2.
Akteure	User
Vorbedingung	Algosim wird ausgeführt.
Fachlicher Auslöser	Der Nutzer möchte seine erlangten Kenntnisse über Algorithmen spielerisch testen.
Normalablauf	<ol style="list-style-type: none"><li>1. <i>Quizseite aufrufen</i></li><li>2. Das System wählt eine Ausgangssituation und einen Algorithmus.</li><li>3. Der Nutzer lässt den Algorithmus ablaufen.</li><li>4. Der Nutzer gibt eine Antwort.</li><li>5. Das System informiert den Nutzer, ob die Antwort korrekt ist.</li><li>6. Der Use-Case ist beendet.</li></ol>
Alternative Abläufe	Siehe Kapitel 9.2.
Nachbedingungen	Der Nutzer hat das Quiz gespielt.

Tabelle 13: Textuelle Beschreibung: Quiz spielen

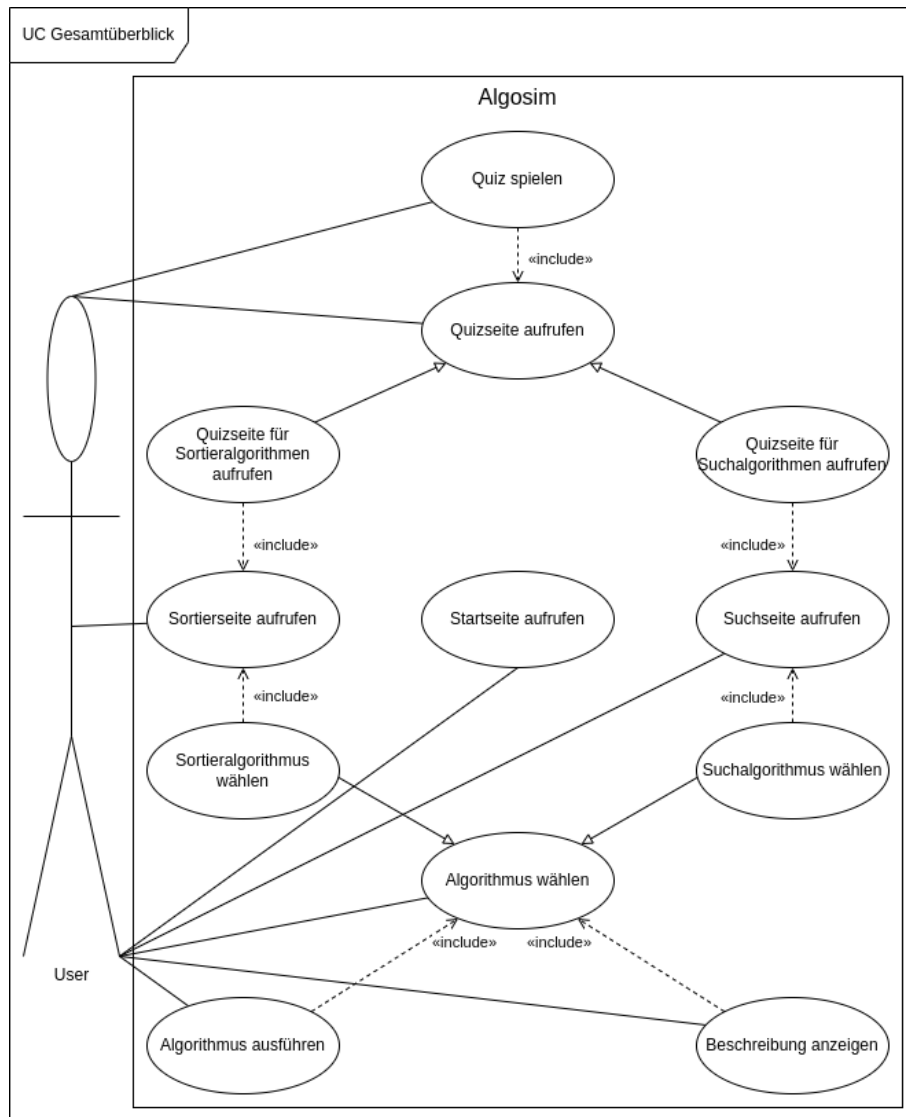


Abbildung 3: Use-Case-Diagramm: Überblick

Name	Quizseite aufrufen
Kurzbeschreibung	Der Nutzer öffnet wahlweise eine der beiden Quizseiten.
Akteure	User
Vorbedingung	Algosim wird ausgeführt.
Fachlicher Auslöser	Der Nutzer möchte eine der beiden Quizseiten öffnen.
Normalablauf	<ol style="list-style-type: none"> <li>1. <i>Sortierseite aufrufen</i></li> <li>2. Der Nutzer klickt auf den Link: „Or try the Quiz“.</li> <li>3. Das System öffnet das Sortierquiz bzw. das Suchquiz.</li> <li>4. Der Use-Case ist beendet.</li> </ol>
Alternative Abläufe	<ol style="list-style-type: none"> <li>1b1. <i>Suchseite aufrufen</i></li> <li>1b2. Weiter mit 2.</li> </ol>
Nachbedingungen	Der Nutzer befindet sich auf der Quizseite.

Tabelle 14: Textuelle Beschreibung: Quizseite aufrufen

Name	Startseite aufrufen
Kurzbeschreibung	Der Nutzer öffnet die Startseite von Algosim, wo er weitere Informationen über das Projekt und die Motivation des Projektteams erlangen kann.
Akteure	User
Vorbedingung	Algosim wird ausgeführt.
Fachlicher Auslöser	Der Nutzer möchte die Startseite aufrufen.
Normalablauf	1. Der Nutzer klickt in der Navigationsleiste der Webseite auf die Schaltfläche „Home“. 2. Das System öffnet die Startseite. 3. Der Use-Case ist beendet.
Alternative Abläufe	/
Nachbedingungen	Der Nutzer befindet sich auf der Startseite.

Tabelle 15: Textuelle Beschreibung: Startseite aufrufen

Name	Sortierseite aufrufen
Kurzbeschreibung	Der Nutzer öffnet die Sortierseite, welche ihm ein Interface zur Gestaltung von Eingabemengen und eine Auswahl an Sortieralgorithmen bietet.
Akteure	User
Vorbedingung	Algosim wird ausgeführt.
Fachlicher Auslöser	Der Nutzer möchte die Sortierseite aufrufen.
Normalablauf	1. Der Nutzer klickt in der Navigationsleiste der Webseite auf die Schaltfläche „Sorting Algorithms“. 2. Das System öffnet die Sortierseite. 3. Der Use-Case ist beendet.
Alternative Abläufe	/
Nachbedingungen	Der Nutzer befindet sich auf der Sortierseite.

Tabelle 16: Textuelle Beschreibung: Sortierseite aufrufen



Name	Suchseite aufrufen
Kurzbeschreibung	Der Nutzer navigiert auf die Seite für Suchalgorithmen. Hier stehen Tools für die Eingabe eines Graphen und den Aufruf verschiedener Suchalgorithmen bereit.
Akteure	User
Vorbedingung	Algosim wird ausgeführt.
Fachlicher Auslöser	Der Nutzer hat ein Interesse für Suchalgorithmen und möchte die Suchseite aufrufen.
Normalablauf	<ol style="list-style-type: none"> <li>1. Der Nutzer klickt in der Navigationsleiste der Webseite auf die Schaltfläche „Search Algorithms“.</li> <li>2. Das System öffnet die Suchseite.</li> <li>3. Der Use-Case ist beendet.</li> </ol>
Alternative Abläufe	/
Nachbedingungen	Der Nutzer befindet sich auf der Suchseite.

Tabelle 17: Textuelle Beschreibung: Suchseite aufrufen

Name	Algorithmus wählen
Kurzbeschreibung	Der Nutzer navigiert auf die Such- oder Sortierseite und wählt einen Algorithmus, mit dem er sich näher beschäftigen möchte.
Akteure	User
Vorbedingung	Algosim wird ausgeführt.
Fachlicher Auslöser	Der Nutzer möchte mehr über einen Algorithmus erfahren.
Normalablauf	<ol style="list-style-type: none"> <li>1. <i>Sortierseite aufrufen</i></li> <li>2. Der Nutzer klickt auf das Drop-Down-Menü mit dem Namen „Algorithm“ / „Select an algorithm“.</li> <li>3. Der Nutzer wählt einen der angezeigten Algorithmen aus und klickt diesen an.</li> <li>4. Das System schließt das Drop-Down-Menü und der Algorithmus ist selektiert.</li> <li>5. Der Use-Case ist beendet.</li> </ol>
Alternative Abläufe	<ol style="list-style-type: none"> <li>1b1. <i>Suchseite aufrufen</i></li> <li>1b2. Weiter mit 2.</li> </ol>
Nachbedingungen	Der Nutzer befindet sich auf der Such- bzw. Sortierseite und hat einen Algorithmus ausgewählt.

Tabelle 18: Textuelle Beschreibung: Algorithmus wählen

Name	Beschreibung anzeigen
Kurzbeschreibung	Der Nutzer hat einen Algorithmus ausgewählt und lässt sich dazu eine erklärende Beschreibung anzeigen.
Akteure	User
Vorbedingung	Algosim wird ausgeführt.
Fachlicher Auslöser	Der Nutzer möchte eine erklärende Beschreibung für einen Algorithmus lesen.
Normalablauf	<ol style="list-style-type: none"> <li>1. <i>Algorithmus wählen</i></li> <li>2. Das System zeigt automatisch eine Beschreibung im unteren Teil der Seite an.</li> <li>3. Der Use-Case ist beendet.</li> </ol>
Alternative Abläufe	/
Nachbedingungen	Dem Nutzer wird eine Beschreibung für einen Algorithmus angezeigt.

Tabelle 19: Textuelle Beschreibung: Beschreibung anzeigen

Name	Algorithmus ausführen
Kurzbeschreibung	Der Nutzer öffnet die Such- oder Sortierseite, wählt einen Algorithmus aus und führt diesen aus. Dabei kann er die zu durchsuchende/sortierende Eingabe selbst bestimmen und den Algorithmus langsamer bzw. schneller abspielen oder ganz pausieren.
Akteure	User
Vorbedingung	Algosim wird ausgeführt.
Fachlicher Auslöser	Der Nutzer möchte einen Algorithmus ausprobieren.
Normalablauf	<ol style="list-style-type: none"> <li>1. <i>Algorithmus wählen</i></li> <li>2. Der Nutzer klickt auf „Generate“, sofern die Schaltfläche noch nicht blau erscheint.</li> <li>3. Der Nutzer wählt die Anzahl der Elemente, sowie den Minimal- und Maximalwert aus.</li> <li>4. Der Nutzer klickt auf „Sort“ bzw. „Search“.</li> <li>5. Das System berechnet den Algorithmus und zeigt anschließend die Ausgangsgrafik an.</li> <li>6. Der Nutzer verwendet die Steuerungsschaltflächen um die Wiedergabe des Algorithmus zu steuern.</li> <li>7. Der Use-Case ist beendet.</li> </ol>

Alternative Abläufe	<p>2a1. Der Nutzer klickt auf „Custom“, sofern die Schaltfläche noch nicht blau erscheint.</p> <p>2a2. Der Nutzer gibt eine durch Kommas getrennte Liste von Zahlen ein.</p> <p>2a3. Weiter mit 4.</p> <p>2b1. Der Nutzer gibt einen Graphen ein und markiert den Start- und Endknoten.</p> <p>2b2. Weiter mit 4.</p> <p>2c1. Der Nutzer klickt auf „Use predefined graph“.</p> <p>2c2. Der Nutzer wählt einen Graphen aus.</p> <p>2c3. Das System zeigt den gewählten Graphen in der Bearbeitungsoberfläche an.</p> <p>2c4. Der Nutzer kann den Graphen weiter bearbeiten.</p> <p>2c5. Weiter mit 4.</p> <p>4a1. Der Nutzer klickt auf „Reset“ (Sortierseite) oder „Clear“ (Suchseite).</p> <p>4a2. Das System setzt alle Eingaben auf die Standardwerte zurück.</p> <p>4a3. Weiter mit 2.</p>
Nachbedingungen	Der Nutzer hat einen Algorithmus ausgeführt.

Tabelle 20: Textuelle Beschreibung: Algorithmus ausführen

## 9.2 Detaildiagramm

Name	Antwort prüfen
Kurzbeschreibung	Nach der Wahl einer Antwort klickt der Benutzer auf die Schaltfläche „Check your answer“ und erfährt, ob seine Antwort korrekt ist.
Akteure	User
Vorbedingung	Algosim wird ausgeführt und der Nutzer befindet sich auf einer Quizseite.
Fachlicher Auslöser	Der Nutzer spielt das Quiz und möchte wissen, ob seine Antwort korrekt ist.
Normalablauf	<ol style="list-style-type: none"> <li>1. <i>Antwort auswählen</i></li> <li>2. Der Nutzer klickt „Check your answer“.</li> <li>3. Das System zeigt an, wie viele Versuche der Nutzer bereits hatte.</li> <li>4. Das System zeigt an, dass die Antwort falsch ist.</li> <li>4. EP: Wiederholung</li> <li>5. EP: Lösung</li> <li>6. Der Use-Case ist beendet.</li> </ol>
Alternative Abläufe	<ol style="list-style-type: none"> <li>4a1. Das System zeigt an, dass die Antwort korrekt ist.</li> <li>4a2. Weiter mit 6.</li> </ol>
Nachbedingungen	Der Nutzer hat eine Quizfrage beantwortet.

Tabelle 21: Textuelle Beschreibung: Antwort prüfen

Name	Korrekte Lösung anzeigen
Kurzbeschreibung	Beim Überprüfen der eingegebenen Lösung erfährt der Nutzer, dass seine Antwort falsch ist. Nachdem er auf die Schaltfläche „Show solution“ klickt, zeigt Algosim die korrekte Antwort.
Akteure	User
Vorbedingung	Die Antwort des Nutzers ist falsch.
Fachlicher Auslöser	Der Nutzer klickt auf die Schaltfläche „Show solution“.
Normalablauf	<ol style="list-style-type: none"> <li>1. <i>Antwort prüfen</i>: EP Lösung</li> <li>2. Das System zeigt die richtige Lösung an.</li> <li>3. Der Use-Case ist beendet.</li> </ol> <p>Weiter mit <i>Antwort prüfen</i> — 6.</p>
Alternative Abläufe	/
Nachbedingungen	Der Nutzer kennt die richtige Antwort.

Tabelle 22: Textuelle Beschreibung: Korrekte Lösung anzeigen

Name	Erneut versuchen
Kurzbeschreibung	Nachdem seine vorherige Antwort falsch ist, entscheidet der Nutzer, dass er einen weiteren Versuch machen möchte, um eine andere Antwort auszuwählen.
Akteure	User
Vorbedingung	Die Antwort des Nutzers ist falsch.
Fachlicher Auslöser	Der Nutzer klickt auf die Schaltfläche „Retry“.
Normalablauf	1. <i>Antwort prüfen</i> : EP Wiederholung 2. Die Frage wird wieder geöffnet. 3. Der Use-Case ist beendet. Weiter mit <i>Antwort prüfen</i> — 1.
Alternative Abläufe	/
Nachbedingungen	Die Nutzer kann eine weitere Antwort versuchen.

Tabelle 23: Textuelle Beschreibung: Erneut versuchen

Name	Neue Frage generieren
Kurzbeschreibung	Nach dem Beantworten einer Frage spielt der Nutzer eine weitere Quizfrage.
Akteure	User
Vorbedingung	Algosim wird ausgeführt und der Nutzer befindet sich auf einer Quizseite.
Fachlicher Auslöser	Der Nutzer möchte nach einer Frage noch eine weitere spielen.
Normalablauf	1. <i>Antwort prüfen</i> 2. Der Nutzer klickt auf „New question“. 2. Das System wählt zufällig einen neuen Algorithmus und eine neue Ausgangssituation aus. 3. Das System setzt die Anzahl der versuchten Antworten zurück. 4. Das System zeigt die nächste Frage an. 5. Der Use-Case ist beendet.
Alternative Abläufe	/
Nachbedingungen	Der Nutzer hat eine neue Frage erhalten.

Tabelle 24: Textuelle Beschreibung: Neue Frage generieren

Name	Antwort auswählen
Kurzbeschreibung	Der Nutzer kann den aktuellen Algorithmus betrachten und wählt aus welchen er vermutet.
Akteure	User
Vorbedingung	Algosim wird ausgeführt und der Nutzer befindet sich auf einer Quizseite.
Fachlicher Auslöser	Der Nutzer möchte die Quizfrage beantworten.
Normalablauf	<ol style="list-style-type: none"> <li>1. Der Nutzer klickt auf „Select the algorithm you guess is correct“.</li> <li>2. Das System öffnet ein Drop-Down-Menü mit allen Such- bzw. Sortieralgorithmen, welche von Algosim unterstützt werden.</li> <li>3. Der Nutzer klickt auf den Algorithmus, den er als Antwort geben möchte.</li> <li>4. Das Drop-Down-Menü wird geschlossen und die Antwort ist selektiert.</li> <li>5. Der Use-Case ist beendet.</li> </ol>
Alternative Abläufe	/
Nachbedingungen	Der Nutzer hat eine Antwort selektiert.

Tabelle 25: Textuelle Beschreibung: Antwort auswählen

Name	Algorithmus ausführen
Kurzbeschreibung	Der Nutzer kann den Algorithmus abspielen lassen und einzelne Schritte genauer inspizieren.
Akteure	User
Vorbedingung	Algosim wird ausgeführt und der Nutzer befindet sich auf einer Quizseite.
Fachlicher Auslöser	Der Nutzer möchte den Algorithmus betrachten.
Normalablauf	<ol style="list-style-type: none"> <li>1. Der Nutzer verwendet die Steuerungsschaltflächen, um die Wiedergabe des Algorithmus zu steuern.</li> <li>2. Der Use-Case ist beendet.</li> </ol>
Alternative Abläufe	/
Nachbedingungen	Der Nutzer hat den Algorithmus inspiziert.

Tabelle 26: Textuelle Beschreibung: Algorithmus ausführen

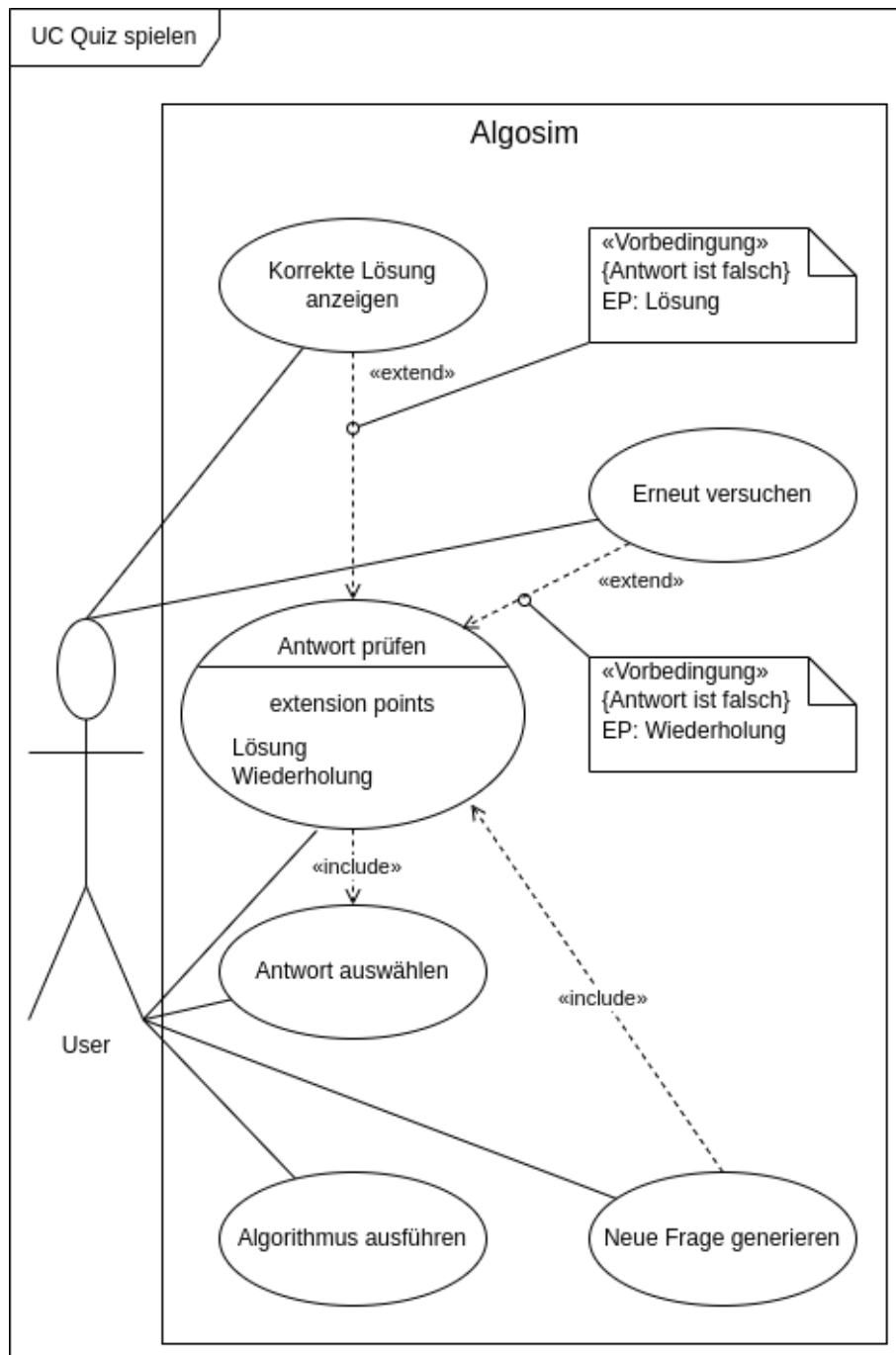


Abbildung 4: Use-Case-Diagramm: Detailansicht

## 10 UML: Aktivitätsdiagramm

Das folgende Aktivitätsdiagramm bezieht sich auf das Vorgehen von Algosim nach dem Auswählen eines Suchalgorithmus. Die dargestellten Aktivitäten geben einen Überblick und sind zu Teilen stark abstrahiert, damit die Übersichtlichkeit des Diagramms gegeben bleibt.

Die Aktivität beginnt damit, dass der eingegebene Graph und der Name des zu verwendenden Algorithmus übergeben werden. Anschließend wird der eingegebene Graph validiert. Wenn die Prüfung Fehler innerhalb des Graphen findet, kommt es zum Aktivitätsende. Falls die Prüfung feststellt, dass der Graph valide ist, wird der GraphConverter damit beauftragt, den Graphen zu einer GraphForm zu konvertieren. Die SearchAlgorithmFactory kann im Anschluss aus der GraphForm und dem Algorithmusnamen eine Algorithmusinstanz erstellen und diese anstoßen. Abschließend wird der jeweils gewählte Algorithmus ausgeführt und das Visualisierungsergebnis von der Aktivität zurückgegeben.

Im darauf folgenden Aktivitätsdiagramm wird die Breitensuche als eine der zuvor dargestellten Aktivitäten genauer ausgeführt. Damit das Aktivitätsdiagramm in seiner Größe nicht mehr als eine DIN-A4-Seite beansprucht, wurde dieser Teil in ein zweites Aktivitätsdiagramm ausgelagert.

Bei der Ausführung von BFS wird zuerst die Knoten-Warteschlange initialisiert, indem der Startknoten in diese eingefügt wird. Anschließend wird, solange die Warteschlange nicht leer ist, der erste Knoten aus der Warteschlange entnommen, damit dieser als besucht markiert werden kann. Wenn es sich um den Endknoten handelt, wird die Schleife abgebrochen, andernfalls werden die noch nicht besuchten Nachbarknoten ermittelt. Der Vorgänger der Nachbarknoten wird entsprechend aktualisiert und die Nachbarknoten werden in die Warteschlange eingereiht. Sobald die Warteschlange leer ist oder der Endknoten erreicht wurde, wird die Visualisierung gebaut und zurückgegeben.



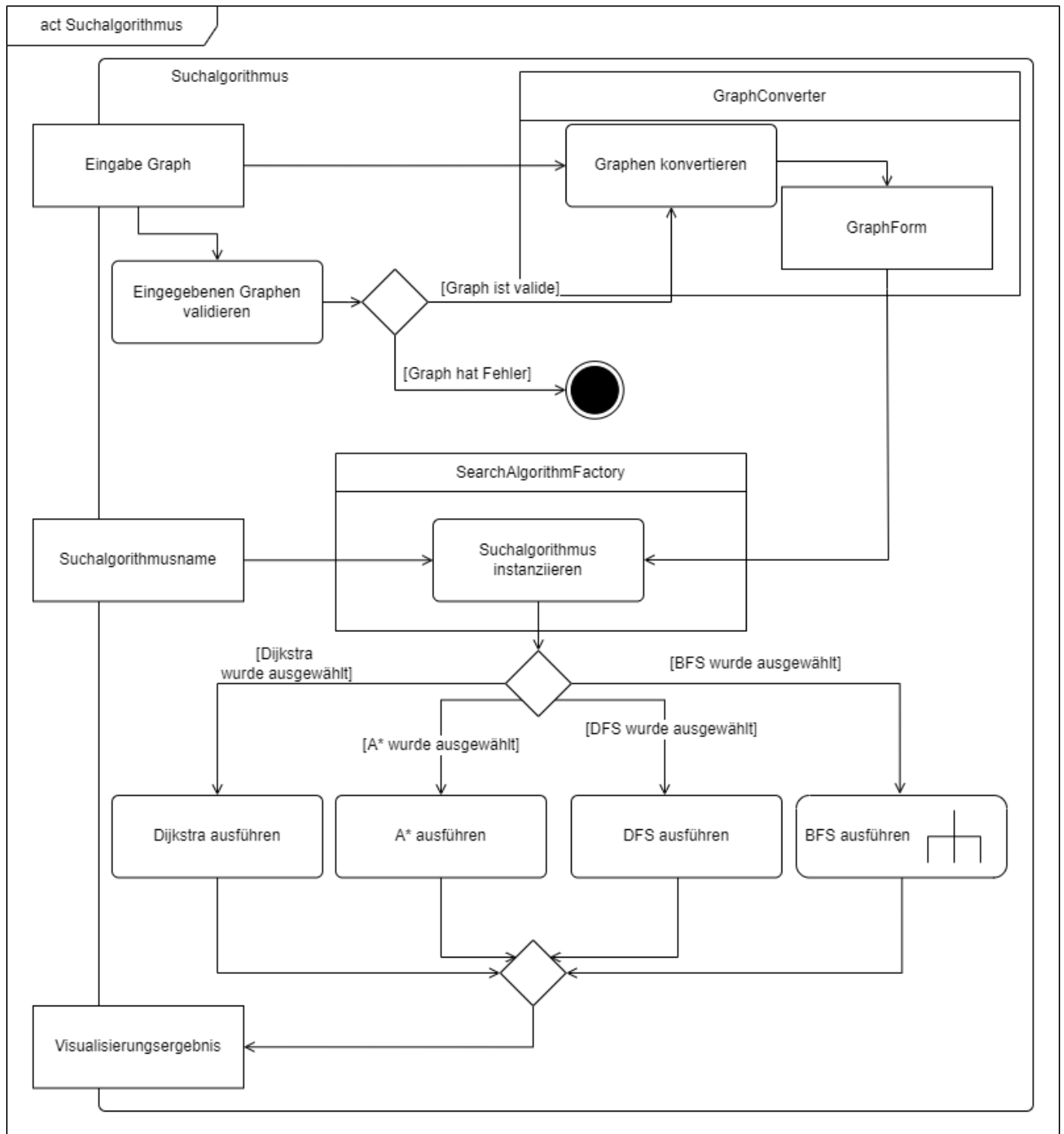


Abbildung 5: Aktivitätsdiagramm

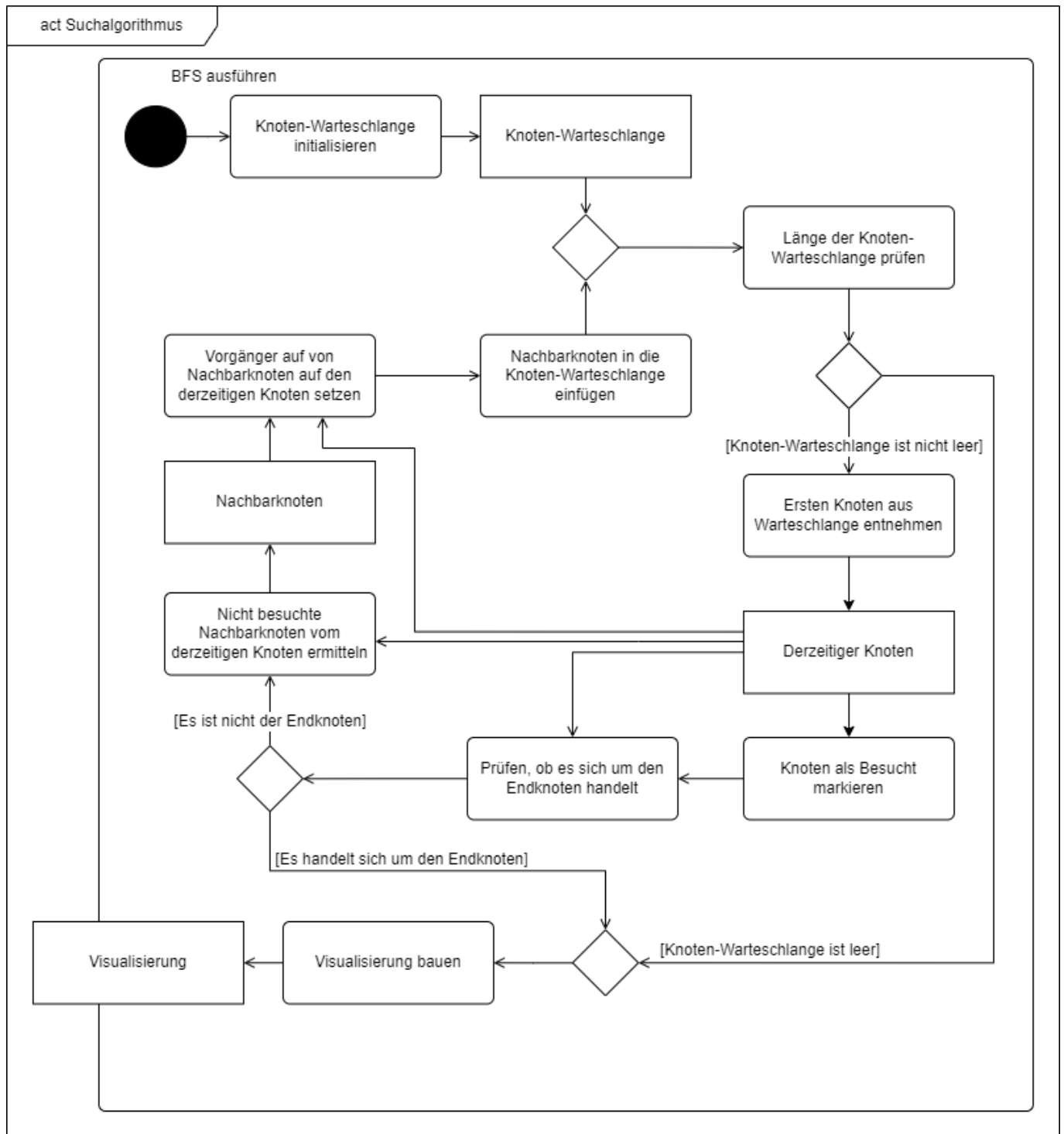


Abbildung 6: BFS ausführen – Teil des Aktivitätsdiagramms

# 11 Qualitätskriterien

Um zu überprüfen, ob das Endprodukt eines Projekts den Vorstellungen gerecht wird, müssen diese zunächst klar definiert werden. Dafür wird ein sogenanntes Qualitätsregister erstellt. Die darin gelisteten Anforderungen bestimmen genau, was das Produkt können muss, wie die Überprüfung stattfindet und in welchem Toleranzbereich das Ergebnis akzeptiert wird.

Jede Anforderung besteht aus einer eindeutigen Qualitätsprüfnummer und bezieht sich auf ein Produkt, welches durch die Produktnummer und den -titel identifiziert wird. Dazu kommt eine Prüfmethode sowie eine genaue Qualitätsprüfbeschreibung, die erklären, wie das Produkt zu kontrollieren ist. Gegebenenfalls kann dabei ein Toleranzbereich spezifiziert werden, der das Produkt auch mit geringen Abweichungen noch als anforderungsgemäß akzeptiert. Für jedes Qualitätskriterium wird außerdem ein verantwortliches Teammitglied bestimmt, welches die Qualitätskontrolle überwacht. Zusätzlich müssen eine oder mehrere Personen mit der Durchführung beauftragt werden. Abschließend enthält das Qualitätsregister einen Plan-Termin für die Durchführung. Hier werden später auch das Ist-Datum der Durchführung sowie das Ergebnis der Kontrolle notiert.

Auch für Algosim hat das Projektteam ein Qualitätsregister erarbeitet. Dieses steht im Anhang B (Seite 39) bereit.

## 12 Technologien und Produkte

Bei der Erarbeitung eines Projekts ist es hilfreich, unterstützende Software zu nutzen, welche die Arbeit erleichtert. Auch im Rahmen der Bearbeitung dieses Projekts kamen solche Technologien und Produkte zum Einsatz.

Die wichtigste verwendete Technologie ist Git. Bei Git handelt es sich um ein System zur Versionsverwaltung (Version Control System oder VCS). Im Rahmen des Projekts wird Git dafür verwendet, das gleichzeitige Arbeiten an der Software zu ermöglichen. Dazu bekommt jedes Feature einen eigenen Versionszweig (Branch), welcher jeweils von einem Teammitglied bearbeitet wird. Nach der vollständigen Implementation des Features wird dieser Versionszweig in den Hauptzweig überführt (merge).

Bei der Integration von Git in den Arbeitsfluss des Projektteams hat sich auch GitHub als zentral erwiesen. GitHub erlaubt es Nutzern, Git-Repositories auf ihren Servern anzulegen. Da diese Dienstleistung kostenlos angeboten wird, ist das Produkt für die Erarbeitung eines Projekts im Rahmen des Studiums optimal. Neben GitHub gibt es auch ähnliche Unternehmen und Organisationen, die solche Dienstleistungen anbieten (z. B. Gitlab). Diese sind jedoch meist kostenpflichtig. Das Projektteam hat sich außerdem für GitHub entschieden, weil die Nutzung sehr einfach ist und jedes Teammitglied bereits Erfahrung im Umgang mit GitHub hatte.

Ein weiteres Produkt, welches bei der Entwicklung von Algosim den Arbeitsaufwand deutlich reduziert hat, ist die vom jeweiligen Entwickler genutzte Entwicklungsumgebung. Nicht alle Gruppenmitglieder entschieden sich dabei für die gleiche IDE. Die meisten Gruppenmitglieder nutzten IntelliJ IDEA Ultimate, da die Lizenz für Studenten kostenlos ist. Außerdem unterstützt IntelliJ die Entwicklung durch syntaktische Hinweise, automatische Vervollständigung und weitere nützliche Funktionen. Einige wenige Teammitglieder haben sich jedoch auch für andere Entwicklungsumgebungen mit ähnlichem Produktumfang entschieden. (z. B. Visual Studio Code)

Für die Entwicklung eines Softwareprodukts wie Algosim ist die Wahl der Programmiersprache sehr entscheidend. Im Rahmen dieses Projekts fiel die Wahl auf TypeScript, da diese Sprache für das gewählte Anwendungsgebiet eine Reihe an Vorteilen bietet. Algosim ist eine Webapplikation, welche nur aus einem Frontend besteht, das bedeutet, dass jeglicher Code im Browser des Anwenders ausgeführt werden

muss. Somit muss die Sprache zu JavaScript kompiliert werden können. TypeScript basiert auf JavaScript und erweitert dieses lediglich um statische Typisierung. Das bedeutet, dass der Typ einer Variable oder eines Feldes nicht mehr zur Laufzeit ermittelt wird, wodurch unerwartetes Verhalten durch die falsche Zuweisung eines Typs verhindert wird. Ein weiterer Punkt bei der Entscheidung für TypeScript war, dass bereits mehrere Gruppenmitglieder mit dieser Programmiersprache gearbeitet hatten.

Es ist nicht sinnvoll, alle benötigten Funktionen selbst zu implementieren, stattdessen sollten möglichst Bibliotheken und Frameworks verwendet werden. Für Algosim werden deshalb eine Reihe von Bibliotheken und ein Framework verwendet. Vue.js ist ein JavaScript-Framework, welches die Entwicklung von Single-Page-Anwendungen vereinfacht. Außerdem bietet Vue.js die Möglichkeit, mit Komponenten zu arbeiten und hilft somit dabei, das Softwareprodukt zu modularisieren. Da diese Anwendungsfälle auf dieses Projekt zutreffen, wird Vue.js genutzt. Weiterführend nutzt Algosim eine Komponentenbibliothek für Vue.js mit dem Namen PrimeVue. Diese Bibliothek implementiert bereits eine Vielzahl an häufig verwendeten Komponenten für Vue.js und erspart dem Projektteam somit die erneute Entwicklung ähnlicher Komponenten.

Damit die Webapplikation ansehnlich aussieht, wird zusätzlich das CSS-Framework Tailwind CSS verwendet. Tailwind enthält eine große Anzahl an CSS-Klassen, welche bei der Entwicklung der grafischen Benutzeroberfläche genutzt werden können.

# 13 Verwendungsanleitung Lösung

Im Hinblick darauf, dass die Nutzer von Algosim verschiedene Voraussetzungen bezüglich ihrer Laufzeitumgebung und Technikerfahrung haben, hat sich das Projektteam dazu entschieden, Algosim auf mehrere Arten ausführbar zu machen. Einige der in diesem Abschnitt beschriebenen Befehle sehen unter Nicht-Windows-Systemen anders aus. Die Ausführung als Docker-Container wird nur für Anwender empfohlen, welche bereits Kenntnisse im Umgang mit Docker, Docker-Compose und Docker-Images haben.

## 13.1 Webseite

Die einfachste Methode, Algosim auszuprobieren, ist das Aufrufen der vom Projektteam bereitgestellten Webseite unter <https://algosim.onrender.com/>. Dieser Aufruf kann mit einem frei gewähltem Browser ausgeführt werden, es wird jedoch Firefox, Google Chrome oder Microsoft Edge empfohlen. Der Webserver wird aber nur für begrenzte Zeit bereitgestellt, weshalb auch andere Methoden zum Ausführen von Algosim ermöglicht werden.

## 13.2 Quellcode

Für diesen Zweck steht der Quellcode des Projekts zur Verfügung. Der Quellcode lässt sich der `Source Code.zip` Datei entnehmen, diese ist unter anderem am GitHub-Release unter <https://github.com/yxyx-github/algosim/releases/tag/v.1.0.1> angehängt. Alternativ kann der Quellcode aber auch in der Abgabe im Moodle-Portal gefunden werden oder direkt mittels Git durch den Befehl `git clone --depth 1 --branch v.1.0.1 https://github.com/yxyx-github/algosim.git` geklont werden. Nach dem Entpacken des Quellcodes muss in den Ordner `algosim` navigiert werden. Zunächst muss nun Node.js installiert werden, sofern dieses noch nicht auf der Maschine installiert ist. Bei Unsicherheit, ob Node.js bereits installiert ist, kann unter Windows der Befehl `npm --version` ausgeführt werden. Sofern Node.js installiert ist und die Umgebungsvariablen korrekt gesetzt wurden, wird durch diesen Befehl die Version von Node.js ausgegeben. Binaries zur Installation von Node.js lassen sich unter <https://nodejs.org/de/download> herunterladen. Anschließend kann der Befehl `npm install` im `algosim`-Ordner ausgeführt werden, um die Dependencies von Algosim zu installieren.

Es gibt zwei verschiedene Möglichkeiten, um mit diesem Quellcode einen Webserver zu starten. Die erste Möglichkeit ist das Starten des Development-Webserver von Node.js mit dem Befehl `npm run dev`. Anschließend sollte die Webseite unter <http://localhost:5173/> erreichbar sein. Auch hier wird die Nutzung von Google Chrome, Microsoft Edge oder Firefox empfohlen.

Die zweite Möglichkeit ist das Kompilieren von Algosim und die Verwendung eines eigenen Webserver. Hierzu muss der Befehl `npm run build` ausgeführt werden. Das Kompilat befindet sich anschließend im `dist`-Verzeichnis. Die Wahl des Servers, welcher das `dist`-Verzeichnis verteilt, wird dem Benutzer überlassen. Das Entwicklungsteam empfiehlt jedoch die Nutzung von `httpd`.

Falls der Webserver als Container gestartet werden soll, wird eine `Dockerfile` und eine `docker-compose.yaml` Datei im Quellcode bereitgestellt. Um diese auszuführen, muss zuerst Docker installiert werden. Das Entwicklerteam empfiehlt die Nutzung von Docker-Desktop, für welches Binaries unter <https://www.docker.com/products/docker-desktop/> verfügbar sind. Es ist dabei zu beachten, dass Virtualisierung im BIOS des Anwendersystems aktiviert sein muss und das Docker auf Windows eine WSL benötigt. Wenn Docker erfolgreich installiert wurde, kann der Befehl `docker --version` ausgeführt werden. Zum Starten der `docker-compose.yaml` muss nun lediglich der Befehl `docker compose up --build` bzw. `docker-compose up --build` ausgeführt werden. Der Webserver sollte anschließend unter <http://localhost:80/> erreichbar sein, sofern der Port 80 nicht blockiert wird. Um die `docker-compose.yaml` wieder abzuräumen, muss lediglich `docker compose down` bzw. `docker-compose down` ausgeführt werden.

## 13.3 Docker-Compose

Falls der Benutzer das Ziel hat, einen eigenen Webserver mittels Docker auszuführen, muss dieser das Image nicht zwingend selbst bauen. Das Entwicklungsteam hat dafür eine `Image Docker Deployment.zip` Datei im GitHub-Release unter <https://github.com/yxyx-github/algosim/releases/tag/v.1.0.1> angehängt. Im Moodle-Portal wurde stattdessen das `Image Docker Deployment.tar.bz2`-Archiv bereitgestellt, da das Zip-Archiv die erlaubten 50 MB überschreitet. Das Archiv muss entpackt werden und es ist in das `algosim`-Verzeichnis zu navigieren. Der `tar -xvjf Image\ Docker\ Deployment.tar.bz2` Befehl kann genutzt werden, um das bz2 komprimierte tar-Archiv zu entpacken. Da Windows bz2 standardmäßig nicht unterstützt, bietet es sich unter diesem Betriebssystem an, den Befehl in der Git-Bash auszuführen. Das `algosim`-Verzeichnis enthält die `docker-compose.yaml`

und ein gespeichertes Docker-Image mit dem Namen `algosim-apache.tar`. Das Docker-Image kann mit dem Befehl `docker load -i algosim-apache.tar` geladen werden. Anschließend kann die YAML-Datei mit dem Befehl `docker compose up` bzw. `docker-compose up` gestartet werden. Die Webseite ist dann unter <http://localhost:80/> erreichbar. Mit dem Befehl `docker compose down` bzw. `docker-compose down` werden die Container wieder abgeräumt.

Um das Docker-Image selbst zu bauen, ohne den Quellcode zu kompilieren, muss die `Prebuild Docker Deployment.zip` vom Moodle-Portal oder vom GitHub-Release (<https://github.com/yxyx-github/algosim/releases/tag/v.1.0.1>) heruntergeladen und entpackt werden. Anschließend ist der `algosim`-Ordner zu öffnen. Hier kann nun der Befehl `docker-compose up --build` bzw. `docker compose up --build` ausgeführt werden, um den Webserver unter <http://localhost:80/> erreichbar zu machen. Mit `docker-compose down` oder `docker compose down` können die Container anschließend wieder gelöscht werden.



# Anhang

# A Risikoregister

RNr.	Risikobeschreibung	P-Eintritt	Auswirkung	Art	Typ	Methode	Behandlungsbeschreibung
1	Front-End ist nicht leistungsfähig genug für die flüssige Darstellung der Animationen	gering	- 2 Ph	technologisch	Bedrohung	Akzeptieren	Ruckelnde Animationen akzeptieren, gegebenenfalls Animationen entfernen
2	Front-End ist nicht leistungsfähig genug für die Berechnungen der Algorithmen	gering	- 12 Ph	technologisch	Bedrohung	Eventualplan	Berechnungen aus dem Front-End abtrennen und in ein eigenes Back-End überführen
3	Mitglied fällt kurzfristig (<1 Woche) aus	mittel	- 10 Ph	sozial	Bedrohung	Risiko reduzieren	Ausreichende Pufferzeit einplanen
4	Mitglied fällt längerfristig (>1 Woche) aus	gering	- 30 Ph	sozial	Bedrohung	Eventualplan	Aufgaben möglichst nicht auf eine einzelne Person verteilen und ausreichende Pufferzeit einplanen
5	Graphen für die Suchalgorithmen stehen bereits in fertigen Bibliotheken bereit	hoch	+ 2 Ph	technologisch	Chance	Ablehnen	Fertige Bibliotheken bergen das Folgerisiko, dass sie einen hohen Zeitaufwand für das Customizing erfordern, welcher nicht im Verhältnis zur Zeitersparnis steht
6	Innerhalb der Vorlesung ist mehr Zeit für die Bearbeitung des Projektes gegeben	mittel	+ 50 Ph	sozial	Chance	Ergreifen	Zusätzliche Zeit innerhalb der Vorlesungen ermöglicht es Aufgaben schneller zu beenden und mehr Spielraum gegen Ende des Projektzeitraums zu erhalten
7	Vorübergehender Ausfall von Infrastruktur, die durch Dritte gestellt wird (u. a. Internetverbindung, GitHub)	mittel	- 2 Ph	technologisch	Bedrohung	Akzeptieren	Alle eingesetzten Entwicklungstools können (gegebenenfalls eingeschränkt) ohne Internetverbindung und dritte Dienste genutzt werden, so kann die Arbeit auch während solcher Ausfälle fortgesetzt werden
8	Ausfall von eigener Infrastruktur (u. a. Computer)	gering	- 2 Ph	technologisch	Bedrohung	Eventualplan	Besorgung und Verwendung von anderen Geräten

# B Qualitätsregister

Nr.	PNr.	PTitel	Qualitäts- prüfmethode	Qualitätsbeschreibung	Toleranz	Verant- wortlich	Durch- führung	Plan- Termin	Ist- Termin	Status / Ergebnis
1	1	Algosim	Funktionsprüfung	Visualisierung von Sortieralgorithmen und Suchalgorithmen		Philip	alle	21.10.	23.10.	erfolgreich abgeschlossen
2	1.1	Algorithmen	Unittests	Testen auf korrekte Ausführung und Wiederholbarkeit		Philip	auto-matisch	06.10	06.10.	erfolgreich abgeschlossen
3	1.1.1	Sortieralgorithmen	Unittests	Testen auf korrekte Ausführung und Wiederholbarkeit	300 Elemente in unter 2 Minuten	Philip	auto-matisch	22.09.	25.09.	erfolgreich abgeschlossen
4	1.1.2	Suchalgorithmen	Unittests	Testen auf korrekte Ausführung und Wiederholbarkeit		Philip	auto-matisch	06.10	06.10.	erfolgreich abgeschlossen
5	1.2	Visualisierung	Funktionsprüfung	Sowohl Suchalgorithmen als auch Sortieralgorithmen können dargestellt werden		Philip	Philip, Marvin	22.09.	03.10.	erfolgreich abgeschlossen
6	1.2.1	Integer als Säulendiagramm	Funktionsprüfung	Testen auf korrekte Darstellung	nicht ganzzahlige Werte müssen nicht unterstützt werden	Philip	Philip, Marvin	28.08.	28.08.	erfolgreich abgeschlossen
7	1.2.2	Graphen darstellen	Funktionsprüfung	Testen auf korrekte Darstellung	Graphen die nicht planar sind, müssen nicht berücksichtigt werden	Marvin	Philip, Marvin	22.09.	03.10.	erfolgreich abgeschlossen
8	1.3	Benutzeroberfläche	Funktionsprüfung	Alle Teileiten sind erreichbar und vollständig.		Nick	Nick	21.10.	23.10.	erfolgreich abgeschlossen
9	1.3.1	Startseite	Korrekturlesen	Der dargestellte Text ist inhaltlich und sprachlich korrekt.		Nick	Nick	21.10.	23.10.	erfolgreich abgeschlossen
10	1.3.2	Sortierseite	Funktionsprüfung	Bedienung muss intuitiv funktionieren.		Philip	Philip	22.09.	25.09.	erfolgreich abgeschlossen
11	1.3.2.1	Algorithmenauswahl	Funktionsprüfung	Testen, dass nach der Auswahl der richtige Algorithmus ausgeführt wird		Marvin	Marvin	31.08.	31.08.	erfolgreich abgeschlossen
12	1.3.2.2	Visualisierung einbinden	Funktionsprüfung	Die Algorithmen werden richtig dargestellt.	Geschwindigkeit der Wiedergabe muss nicht stimmen	Marvin	Marvin	30.08.	30.08.	erfolgreich abgeschlossen
13	1.3.2.3	Beschreibung der Algorithmen	Korrekturlesen	Die Beschreibungen werden vollständig dargestellt. Die Texte sind fehlerfrei formuliert. Der Inhalt der Beschreibungen entspricht der Wahrheit.		Paul	Paul	22.09.	06.10.	erfolgreich abgeschlossen
14	1.3.2.4	Eigene Eingaben für Algorithmen	Funktionsprüfung	Testen, dass nach der Auswahl der richtige Algorithmus ausgeführt wird	nicht ganzzahlige Werte müssen nicht unterstützt werden	Mavin	Mavin	31.08.	31.08	erfolgreich abgeschlossen
15	1.3.3	Suchseite	Funktionsprüfung	Bedienung muss intuitiv funktionieren.		Philip	Philip	06.10.	06.10.	erfolgreich abgeschlossen
16	1.3.3.1	Algorithmenauswahl	Funktionsprüfung	Testen, dass nach der Auswahl der richtige Algorithmus ausgeführt wird		Marvin	Marvin	22.09.	03.10.	erfolgreich abgeschlossen
17	1.3.3.2	editierbarer Graph	Funktionsprüfung	Der Graph ist über die Nutzeroberfläche editierbar. Die Datenstruktur wird entsprechend angepasst.	Graphen die nicht planar sind, müssen nicht berücksichtigt werden	Marvin	Marvin	22.09.	03.10.	erfolgreich abgeschlossen
18	1.3.3.3	Visualisierung einbinden	Funktionsprüfung	Die Algorithmen und Graphen werden richtig dargestellt.	Graphen die nicht planar sind, müssen nicht berücksichtigt werden	Marvin	Marvin	22.09.	03.10.	erfolgreich abgeschlossen
19	1.3.3.4	Beschreibung der Algorithmen	Korrekturlesen	Die Beschreibungen werden vollständig dargestellt. Die Texte sind fehlerfrei formuliert. Der Inhalt der Beschreibungen entspricht der Wahrheit		Onur	Onur	06.10.	06.10.	erfolgreich abgeschlossen

20	1.3.4	Quiz	Funktionsprüfung	Die Algorithmenwahl ist zufällig. Die Antworten werden richtig bearbeitet.		Onur	Onur	21.10.	21.10.	erfolgreich abgeschlossen
21	1.3.4.1	Visualisierung einbinden	Funktionsprüfung	Die Algorithmen werden richtig dargestellt		Onur	Onur	13.10.	21.10.	erfolgreich abgeschlossen
22	1.3.4.2	Antworten	Funktionsprüfung	Es wird eine Auswahl an Antwortmöglichkeiten gegeben		Onur	Onur	13.10.	21.10.	erfolgreich abgeschlossen
23	1.3.4.3	Ergebnisauswertung	Funktionsprüfung	Das Ergebnis entspricht der Wahrheit		Onur	Onur	21.10.	21.10.	erfolgreich abgeschlossen
24	1.3.5	Logo	Visuelle Prüfung	Das Logo sollte ansprechend sein. Das Logo darf keine Urheberrechte oder Markenrechte verletzen.		Nick	Nick	29.09.	23.10.	erfolgreich abgeschlossen