IU Internationale Hochschule
Weiterbildung: “Software Engineering - Python”
Modul : DLMCSPSE01_D - Projekt: Software Engineering
Tutor: Prof. Dr. David Kuhlen

Prüfungsleistung: Portfolio

Anforderungsdokument

Eingereicht am 16.09.2025.

Verfasser:
Djahan Bayrami Latran
Denninger Straße 198
81927 München
E-Mail: djahan.latran@gmail.com
Matrikelnummer: UPS10672478

1. Stakeholder:

- **Studenten/Schüler**: Beispielsweise zur Lernunterstützung von Studenten der Informatik, Wirtschaftsinformatik oder ähnlichen Fächern.
- **Interessierte/Quereinsteiger/Programmier-Neulinge**: Personen, die sich für die Thematik interessieren und sich gerne zusätzliches Wissen aneignen wollen.
- **Lehrkräfte**: Die App kann beispielsweise im Unterricht/Vorlesungen von Lehrkräften verwendet werden, um neben der Theorie den Studenten die Funktionsweisen der Algorithmen anschaulich zu vermitteln.

2. Funktionale Anforderungen als User-Stories:

- **FA1: Visualisierung von Algorithmen**
  Als Nutzer möchte ich eine korrekte visuelle Repräsentation der Funktionsweise von Algorithmen, um deren Logik und die Ablaufschritte besser verstehen zu können.
- **FA2: Algorithmik-Grundlagen Vermittlung**
  Als Nutzer möchte ich Algorithmen zur Auswahl haben, die mir grundlegende Algorithmik-Konzepte vermitteln, um das Erlernte auf komplexere Probleme anwenden zu können.
- **FA3: Vertiefung durch Zusatzinformationen**
  Als Nutzer möchte ich zu jedem auswählbaren Algorithmus zusätzliche Informationen in Form von Quellcode mit Kommentaren und einer Beschreibung angezeigt bekommen, um das vermittelte Wissen zu vertiefen.
- **FA4: Interaktive Parameter-Steuerung**
  Als Nutzer möchte ich verschiedene Parameter wie Eingabegröße und Geschwindigkeit der Algorithmus-Schritte verändern können, um auf diese Weise unterschiedliche Szenarien zu simulieren und die Logik besser verstehen zu können.
- **FA5: Interaktive Algorithmus-Steuerung**
  Als Nutzer möchte ich den Ablauf des ausgewählten Algorithmus pausieren und fortsetzen können, um die einzelnen Schritte besser nachvollziehen zu können.

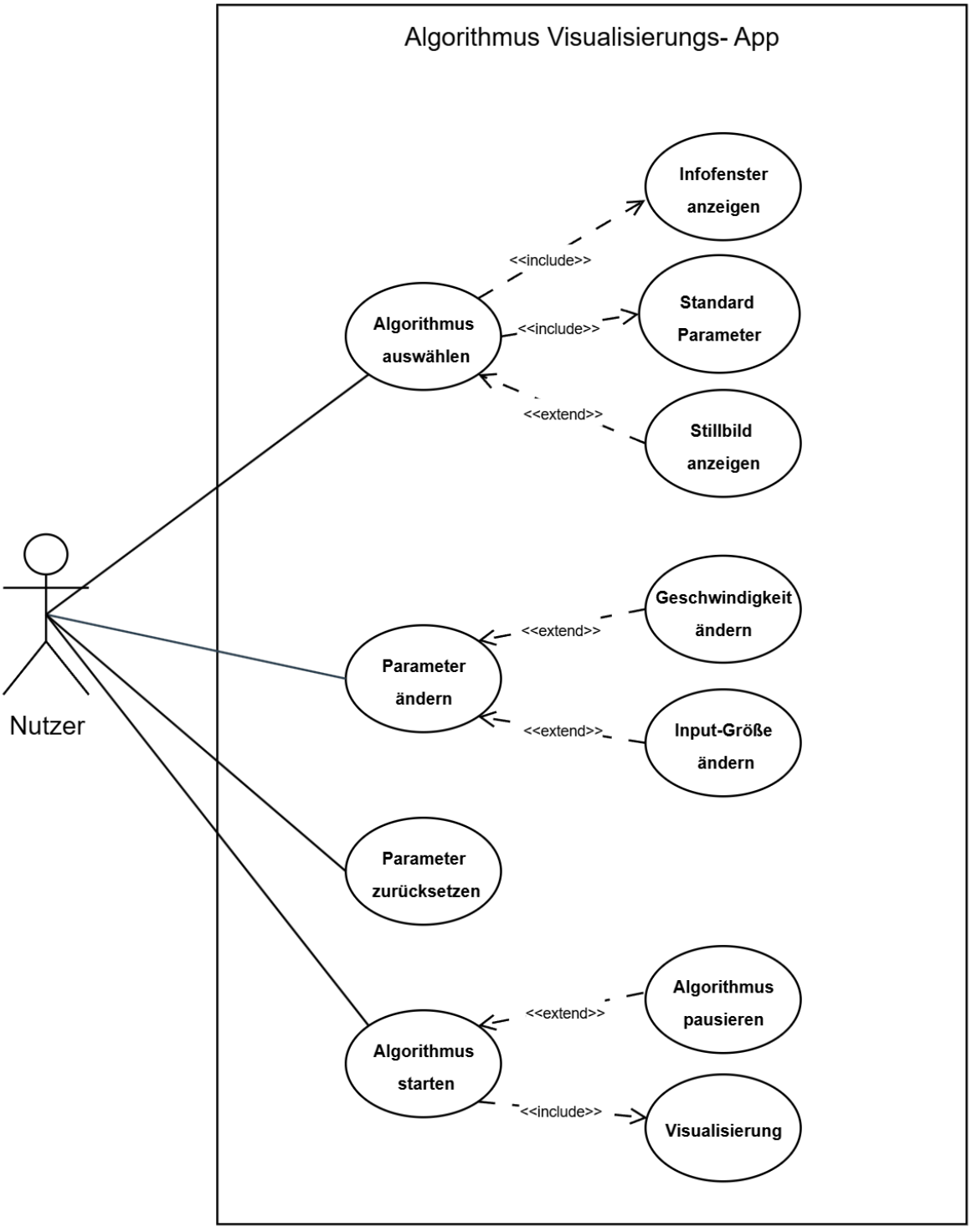

Abbildung 3: Use-Case-Diagramm zur Visualisierung der funktionalen Anforderungen, erstellt mit draw.io

3. Nichtfunktionale Anforderungen:

- **NFA1: Performance der grafischen Animationen**

  Anforderung:

  Die Anwendung muss grafische Animationen, in Form von animierten 2D Balkendiagrammen und Gitternetzen flüssig mit mindestens 30 FPS darstellen können. Auch bei größeren Eingabemengen (bis zu 100 Elemente bei Sortieralgorithmen) muss die Anwendung weiterhin fehlerfrei und performant laufen.

  Erfüllung:

  Es wird eine etablierte und performante Bibliothek verwendet, die für grafische Darstellungen optimiert wurde (z.B. Pygame). Die Bildwiederholungsrate wird regelmäßig mit Leistungstests überwacht und geprüft. Es werden geeignete, performante Datenstrukturen verwendet.

- **NFA2: Benutzerfreundlichkeit**

  Anforderung:

  Das Benutzeroberflächen Design muss intuitiv gestaltet und selbsterklärend navigierbar sein, sodass auf eine zusätzliche Anleitung zur Bedienung der Anwendung verzichtet werden kann.

  Erfüllung:

  Die Gestaltung des User-Interface wird gemäß dem zuvor erarbeiteten 2D-Konzepts ausgeführt. Das Layout wird logisch und klar strukturiert mit einfachen Schaltflächen gestaltet. Es wird ein einheitliches Farbschema mit Akzentfarbe zur Betonung von wichtigen Schaltflächen und zur Navigationsunterstützung verwendet. Usability-Tests werden mit externen Personen durchgeführt.

- **NFA3: Erweiterbarkeit (Modularität)**

  Anforderung:

  Die Anwendung muss so modular strukturiert sein, dass der Aufwand zum Implementieren zusätzlicher Algorithmen oder Darstellungsmethoden kleinstmöglich gehalten wird.

  Erfüllung:

  Während der Planung der Softwarearchitektur wird auf eine lose Kopplung der einzelnen Module geachtet. Jeder in der Applikation auswählbare Algorithmus wird

als eine separate Klasse implementiert, die von einer getrennten Visualisierungs-Komponente dargestellt wird. Die Visualisierungs-Komponente beinhaltet Klassen für die Visualisierungsarten (Darstellung als Balkendiagramme und Gitternetze).
Sollte es zukünftig Bedarf geben, die Applikation um weitere Algorithmen zu ergänzen, können die Algorithmen als einzelne Klassen in das Algorithmus-Modul integriert werden.

- **NFA4: Reaktionsgeschwindigkeit bei Interaktion**
  Anforderung:
  Interaktionen vom Nutzer mit der Anwendungsoberfläche (z.B. Algorithmus-Auswahl, “Play”-Schaltflächen Betätigung, Parameter Änderungen) müssen stets flüssig, ohne Verzögerungen ablaufen.
  Erfüllung:
  Das Animationsfenster und der Rest vom GUI werden in unterschiedlichen/separaten Intervallen aktualisiert, sodass die Geschwindigkeitssteuerung des Algorithmus nicht die Reaktionsgeschwindigkeit der Schaltflächen beeinflusst.

- **NFA5: Windows Kompatibilität**
  Anforderung:
  Die Anwendung muss unter Windows 10/11 lauffähig sein.
  Erfüllung:
  Die Anwendung wird auf einem Windows-Betriebssystem implementiert und getestet. Die verwendeten Bibliotheken sind kompatibel mit Windows.

4. Glossar:

- **Animation**: Veränderung von statischen visuellen/grafischen Elementen über Zeit, um Veränderung und/oder Bewegung darzustellen.
- **Zeitkomplexität**: Die Zeitkomplexität zeigt, wie sich die Laufzeit eines Algorithmus bei sehr großen Eingabemengen verhält.
- **Laufzeit**: Die Laufzeit gibt an, wie lange ein Algorithmus zur Durchführung seiner Schritte benötigt.

- **Eingabe-Input/Eingabemenge**: Veränderliche Daten, die der Algorithmus zu verarbeiten hat.
- **Bildwiederholungsrate**: Die Rate an einzelnen Bildern, die in einem bestimmten Zeitrahmen (meist pro Sekunde) dargestellt werden.