Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Projektarbeit

Internationale Hochschule Duales Studium

Studiengang: Wirtschaftsinformatik

**Erstellung einer Passwortgenerierungssoftware**

Florian Lichtenberger

Matrikelnummer: 102205743

Julia Sadlo

Matrikelnummer: ???

Angelo Vienibene

Matrikelnummer: ???

Betreuende Person: Alexander Büchner

Abgabedatum: 23.07.2024

**Inhaltsverzeichnis**

[**1 Einleitung** 1](#_Toc172293342)

[**2 Vorstellung der Software** 2](#_Toc172293343)

[**3 Qualitätsplanung** 3](#_Toc172293344)

[**4 Vorstellung und Anwendung der Prüftechniken** 5](#_Toc172293345)

[**4.1 Unittests** 5](#_Toc172293346)

[**4.2 Äquivalenzklassen?** 7](#_Toc172293347)

[**4.3 Walkthrough** 7](#_Toc172293348)

[**5 Reflexion** 7](#_Toc172293349)

[**4** **Fazit** 9](#_Toc172293350)

# **1 Einleitung**

In der heutigen digitalen Welt sind sichere Passwörter unerlässlich, um sensible Informationen zu schützen und den Zugriff auf vertrauliche Daten zu sichern. Die Erstellung starker Passwörter ist jedoch oft eine Herausforderung für Benutzer, die einfache und leicht zu merkende Kombinationen bevorzugen (Quelle). Diese Arbeit widmet sich der Entwicklung einer Software zur automatisierten Generierung sicherer Passwörter. Ziel der Software ist es, Benutzern zu helfen, Passwörter zu erstellen, die sowohl komplex als auch benutzerfreundlich sind.

Die Qualität und Sicherheit einer solchen Software sind von entscheidender Bedeutung, weshalb umfassende Qualitätssicherungsmaßnahmen unerlässlich sind. Im Rahmen dieser Arbeit werden verschiedene Qualitätssicherungstools vorgestellt und angewendet, um die Software kontinuierlich zu verbessern und Fehler zu identifizieren sowie zu beheben. Dabei werden sowohl statische als auch dynamische Testmethoden eingesetzt, um sicherzustellen, dass die Software den höchsten Standards entspricht und zuverlässig funktioniert.

Durch die Kombination von innovativer Softwareentwicklung und Qualitätssicherung soll ein robustes Programm entstehen, das den Anforderungen an Sicherheit und Benutzerfreundlichkeit gerecht wird. Diese Arbeit dokumentiert den gesamten Entwicklungsprozess, die angewandten Testmethoden und die erzielten Verbesserungen, um ein umfassendes Bild der Softwareentwicklung und Qualitätssicherung zu vermitteln.

Zunächst werden die Funktionen und Anforderungen der Passwortgenerierungssoftware detailliert vorgestellt. Hierbei wird erläutert, welche Ziele die Software verfolgt, welche Funktionen sie bietet und welche technischen und sicherheitsrelevanten Anforderungen sie erfüllen muss. Im Anschluss daran wird der Qualitätsplan präsentiert, der die Strategien und Methoden zur Sicherstellung der Softwarequalität beschreibt. Dieser Plan bildet die Grundlage für die Auswahl und Anwendung der Qualitätssicherungstools. Daraufhin werden drei spezifische Qualitätssicherungstools im Detail beschrieben und ihre Anwendung auf die Softwareentwicklung erläutert. Diese Tools sind entscheidend, um die Software auf Schwachstellen zu testen und kontinuierliche Verbesserungen vorzunehmen. Im weiteren Verlauf erfolgt eine Reflexion der durchgeführten Qualitätssicherungsmaßnahmen. Es wird analysiert, welche Aspekte des Qualitätssicherungsprozesses gut funktionierten und welche Schwierigkeiten auftraten. Diese Reflexion bietet wertvolle Erkenntnisse für zukünftige Projekte und die Optimierung von Qualitätssicherungsprozessen. Abschließend wird ein Fazit gezogen, das die Ergebnisse der Arbeit zusammenfasst und einen Ausblick auf mögliche Weiterentwicklungen der Software und zukünftige Qualitätssicherungsmaßnahmen gibt.

# **2 Vorstellung der Software**

In dieser Arbeit wird ein Passwortgenerator vorgestellt, der benutzerdefinierte Passwörter erzeugt, um die Sicherheit von Accounts und Daten zu erhöhen. Der Generator wurde als eine Webanwendung implementiert, die es dem Benutzer ermöglicht, verschiedene Parameter zur Generierung von Passwörtern festzulegen. Im Folgenden wird der Code des Passwortgenerators detailliert beschrieben und analysiert.

Die Hauptfunktion der Software ist die Generierung sicherer Passwörter basierend auf den vom Benutzer angegebenen Kriterien. Der Benutzer kann die Länge des Passworts sowie die Verwendung von Großbuchstaben, Kleinbuchstaben, Zahlen und Sonderzeichen festlegen.

Die Software bietet mehrere Parameter, die vom Benutzer eingestellt werden können:

1. **Länge des Passworts**: Der Benutzer kann die gewünschte Länge des Passworts festlegen. Die Länge muss zwischen 12 und 40 Zeichen liegen.
2. **Zeichenarten**: Der Benutzer kann auswählen, welche Arten von Zeichen im Passwort enthalten sein sollen:
   * Großbuchstaben (A-Z)
   * Kleinbuchstaben (a-z)
   * Zahlen (0-9)
   * Sonderzeichen (!@#$%^&\*()\_+[]{}|;:,.<>?)

Die Software hat bestimmte Begrenzungen, die beachtet werden müssen:

* **Minimale und maximale Passwortlänge**: Die Länge des Passworts muss zwischen 12 und 40 Zeichen liegen. Falls die angegebene Länge außerhalb dieses Bereichs liegt, wird der Benutzer darauf hingewiesen und die Passwortgenerierung wird nicht durchgeführt.
* **Zeichenarten-Auswahl**: Mindestens eine Zeichenart muss ausgewählt sein, um ein Passwort generieren zu können. Falls keine Zeichenart ausgewählt wurde, wird ein Fehler ausgelöst.

Die Software wird durch eine einfache Benutzeroberfläche gesteuert, die in HTML und CSS gestaltet ist. Die Passwortgenerierung erfolgt durch JavaScript. Im Folgenden werden die einzelnen Komponenten des Codes beschrieben:

Die HTML-Struktur definiert die Benutzeroberfläche des Passwortgenerators. Hier ein Auszug des HTML-Codes: ?????

Der obige HTML-Code erzeugt eine benutzerfreundliche Oberfläche, in der der Benutzer die Länge des Passworts und die Zeichenarten auswählen kann. Ein Klick auf den Button "Passwort generieren" ruft die generate-Funktion auf.

Die JavaScript-Funktion generate sammelt die Benutzereingaben und ruft die Funktion generatePassword auf, um das Passwort zu erzeugen.

Die Funktion überprüft die Eingaben des Benutzers und stellt sicher, dass die Passwortlänge innerhalb des erlaubten Bereichs liegt. Anschließend werden die Einstellungen zur Generierung des Passworts an die Funktion generatePassword übergeben.

Die Funktion generatePassword erstellt das Passwort basierend auf den Benutzereinstellungen.

Die Funktion generatePassword nimmt fünf Parameter entgegen:

* length: Die gewünschte Länge des Passworts.
* useUppercase: Ein boolescher Wert, der angibt, ob Großbuchstaben verwendet werden sollen.
* useLowercase: Ein boolescher Wert, der angibt, ob Kleinbuchstaben verwendet werden sollen.
* useNumbers: Ein boolescher Wert, der angibt, ob Zahlen verwendet werden sollen.
* useSpecial: Ein boolescher Wert, der angibt, ob Sonderzeichen verwendet werden sollen.

Zunächst werden Zeichenketten für die verschiedenen Zeichenarten definiert. Je nach Benutzereingaben werden diese Zeichenketten in die Variable allChars aufgenommen. Wenn keine Zeichenart ausgewählt wurde, wird ein Fehler ausgelöst. Anschließend wird das Passwort der gewünschten Länge generiert, indem zufällig Zeichen aus allChars ausgewählt werden.

Der Passwortgenerator bietet eine flexible und benutzerfreundliche Möglichkeit, sichere Passwörter zu generieren. Durch die verschiedenen Parameter zur Anpassung der Passwortlänge und -komplexität kann der Benutzer Passwörter erstellen, die den spezifischen Sicherheitsanforderungen entsprechen. Die Implementierung in HTML, CSS und JavaScript ermöglicht eine einfache Nutzung direkt im Browser.

# **3 Qualitätsplanung**

Die Qualitätsplanung ist ein wesentlicher Bestandteil des Softwareentwicklungsprozesses und umfasst alle Aktivitäten, die notwendig sind, um sicherzustellen, dass das Endprodukt den festgelegten Qualitätsstandards und Anforderungen entspricht. Für den Passwortgenerator wurden mehrere Maßnahmen und Strategien definiert, um die Qualität der Software zu gewährleisten.

Das Hauptziel der Qualitätsplanung in diesem Projekt besteht darin, ein zuverlässiges und sicheres Werkzeug zur Passwortgenerierung zu entwickeln. Dies beinhaltet:

* Sicherstellung der korrekten Funktionalität aller Features.
* Gewährleistung einer benutzerfreundlichen Oberfläche.
* Minimierung von Fehlern und Sicherheitslücken.
* Sicherstellung der Einhaltung von Best Practices in der Softwareentwicklung.

Die Qualitätsanforderungen für den Passwortgenerator umfassen sowohl funktionale als auch nicht-funktionale Anforderungen:

1. **Funktionale Anforderungen**:
   * Der Passwortgenerator muss Passwörter generieren, die den vom Benutzer festgelegten Kriterien entsprechen.
   * Die Software muss Eingabefehler korrekt behandeln und den Benutzer entsprechend informieren.
   * Die generierten Passwörter müssen eine ausreichende Komplexität aufweisen, um die Sicherheit zu gewährleisten.
2. **Nicht-funktionale Anforderungen**:
   * Benutzerfreundlichkeit: Die Benutzeroberfläche muss intuitiv und einfach zu bedienen sein.
   * Performance: Die Passwortgenerierung muss schnell und ohne spürbare Verzögerung erfolgen.
   * Sicherheit: Der Code muss sicher sein und darf keine Schwachstellen aufweisen, die zu Sicherheitsrisiken führen könnten.

Die Qualitätsstrategie umfasst mehrere Ansätze und Methoden, die zur Sicherstellung der Qualität des Passwortgenerators eingesetzt werden:

1. **Code-Reviews und Pair-Programming**:
   * Regelmäßige Code-Reviews durch Teammitglieder, um Fehler frühzeitig zu identifizieren und zu beheben.
   * Pair-Programming, um die Qualität des Codes durch gemeinsames Arbeiten an Problemlösungen zu verbessern.
2. **Automatisierte Tests**:
   * Unit-Tests, um sicherzustellen, dass einzelne Funktionen wie die generatePassword-Funktion korrekt arbeiten.
3. **Qualitätssicherungstools**:
   * Verwendung von Qualitätssicherungstools zum Identifizieren von Fehlern bzw. Verbesserungsmöglichkeiten

Der Qualitätsplan für das Projekt umfasst die folgenden Schritte:

1. **Anforderungsanalyse**:
   * Sammlung und Dokumentation der Anforderungen an die Software.
   * Definition von Qualitätsstandards und -zielen.
2. **Planung und Design**:
   * Erstellung eines Designs, das den Qualitätsanforderungen entspricht.
   * Planung der Testfälle und Auswahl der Qualitätssicherungstools.
3. **Implementierung**:
   * Programmierung des Passwortgenerators unter Berücksichtigung der festgelegten Qualitätsrichtlinien.
   * Durchführung von Code-Reviews und Pair-Programming.
4. **Testen und Validierung**:
   * Durchführung von Unit-Tests
   * Verwendung von Qualitätssicherungstools.
5. **Wartung und Verbesserung**:
   * Regelmäßige Überprüfung und Aktualisierung der Software, um neue Sicherheitsanforderungen und Benutzerbedürfnisse zu erfüllen.

# **4 Vorstellung und Anwendung der Prüftechniken**

## **4.1 Unittests**

Im Rahmen der Qualitätssicherung des Passwortgenerators wurden Unit-Tests eingesetzt, um die Funktionalität und Zuverlässigkeit der Software sicherzustellen. Unit-Tests sind eine Methode der Softwareprüfung, bei der einzelne Einheiten oder Funktionen der Software isoliert getestet werden, um sicherzustellen, dass sie erwartungsgemäß funktionieren. Im Folgenden wird erläutert, wie die Unit-Tests für den Passwortgenerator implementiert wurden, welche Software verwendet wurde, und welche Vor- und Nachteile sowie Limitationen mit Unit-Tests verbunden sind.

**Unit-Tests** sind automatisierte Tests, die spezifische Teile des Codes in Isolation überprüfen. Im Fall des Passwortgenerators wurden Unit-Tests geschrieben, um verschiedene Aspekte der Funktion generatePassword zu testen. Die Tests sind in der Datei password-generator.spec.js definiert und verwenden das Testing-Framework Jasmine.

**Code-Schnipsel aus password-generator.spec.js: ???**

Für das Unit-Testing wurde **Jasmine** verwendet, ein populäres JavaScript-Testing-Framework, das speziell für das Testen von JavaScript-Code entwickelt wurde. Jasmine ermöglicht es, Tests in einer klaren, beschreibenden Syntax zu schreiben und bietet Funktionen wie describe, it, und expect zur Definition und Auswertung von Testfällen.

**Jasmine Integration im HTML-Dokument: ???**

1. **Früherkennung von Fehlern**: Unit-Tests helfen, Fehler frühzeitig zu erkennen, da einzelne Funktionen isoliert getestet werden. Dies reduziert die Wahrscheinlichkeit, dass Fehler in späteren Phasen des Entwicklungsprozesses übersehen werden.
2. **Schnelle Fehlersuche**: Bei fehlerhaften Tests können die Probleme schnell lokalisiert und behoben werden, da die Tests auf spezifische Funktionalitäten abzielen.
3. **Dokumentation des Codes**: Unit-Tests dienen als Dokumentation für die erwartete Funktionalität eines Codes. Neue Entwickler können die Tests nutzen, um zu verstehen, wie die Software funktionieren soll.
4. **Verhinderung von Regressionen**: Durch das regelmäßige Ausführen von Unit-Tests kann sichergestellt werden, dass neue Änderungen im Code keine bestehenden Funktionen beschädigen.
5. **Begrenzte Testabdeckung**: Unit-Tests überprüfen nur einzelne Funktionen und können keine Fehler aufdecken, die durch die Interaktion zwischen verschiedenen Modulen oder Systemen entstehen.
6. **Initialer Aufwand**: Das Schreiben von Unit-Tests kann anfangs zeitaufwendig sein und erfordert eine zusätzliche Planungs- und Implementierungsphase.
7. **Wartungsaufwand**: Bei Änderungen am Code müssen möglicherweise auch die zugehörigen Tests angepasst werden, was zusätzlichen Wartungsaufwand verursacht.
8. **Keine vollständige Abdeckung von Integration und Systemtests**: Unit-Tests decken nur die Funktionen der einzelnen Einheiten ab und überprüfen nicht die Integration oder das Gesamtsystemverhalten.
9. **Unzureichende Überprüfung von Benutzeroberflächen**: Unit-Tests konzentrieren sich auf den Backend-Code und können keine Benutzeroberflächenprobleme oder Usability-Probleme erkennen.
10. **Schwierigkeiten bei der Simulation von komplexen Abhängigkeiten**: Tests, die auf externe Systeme oder komplexe Abhängigkeiten angewiesen sind, können schwierig zu schreiben und zu pflegen sein.

Für das Passwortgenerator-Projekt haben Unit-Tests eine große Rolle gespielt. Sie ermöglichten es, die Kernfunktionalität der Passwortgenerierung zu überprüfen, einschließlich der Länge des Passworts und der Einbeziehung der verschiedenen Zeichenarten. Die Tests halfen dabei, sicherzustellen, dass die Software die Anforderungen korrekt umsetzt und dass Änderungen am Code keine unerwünschten Nebenwirkungen haben.

Durch die regelmäßige Durchführung der Unit-Tests konnte das Projektteam sicherstellen, dass die Software robust und zuverlässig ist. Auch wenn Unit-Tests nicht alle Arten von Fehlern abdecken, trugen sie wesentlich dazu bei, die Qualität der Software zu sichern und die Entwicklungsprozesse zu optimieren. Für zukünftige Projekte wird empfohlen, die Unit-Tests frühzeitig im Entwicklungsprozess zu integrieren und mit anderen Testmethoden wie Integrationstests zu kombinieren, um eine umfassendere Testabdeckung zu gewährleisten.

## **4.2 Äquivalenzklassen?**

## **4.3 Walkthrough**

# **5 Reflexion**

Im Rahmen des Projekts zur Entwicklung des Passwortgenerators wurden sowohl positive Aspekte als auch Herausforderungen identifiziert. Die Reflexion bietet eine kritische Betrachtung der Erfahrungen und leitet Empfehlungen für zukünftige Projekte ab.

**Projekt- und Arbeitsaufteilung**: Die Aufteilung des Projekts und die Zuordnung der Aufgaben innerhalb des Teams waren effektiv. Die Aufgaben wurden klar definiert, und die Verantwortlichkeiten waren gut verteilt. Dies trug maßgeblich zur effizienten Umsetzung des Projekts bei und stellte sicher, dass jeder Teammitglied seine Stärken einbringen konnte.

**Teamarbeit**: Die Zusammenarbeit im Team war durchweg positiv. Regelmäßige Meetings und eine offene Kommunikation förderten einen effektiven Austausch von Ideen und die Lösung von Problemen. Das Pair-Programming und die Code-Reviews trugen zur Qualität des Codes bei und ermöglichten eine schnelle Identifikation und Behebung von Fehlern.

**Coding-Qualität**: Der Code wurde gut implementiert und erfüllt die grundlegenden Anforderungen des Projekts. Die verwendeten Programmiersprachen und Technologien erwiesen sich als geeignet, und die Passwortgenerierungslogik funktionierte zuverlässig. Die Einhaltung der Coding-Standards und der saubere Code trugen zur Stabilität der Software bei.

**Definition der Anforderungen**: Die Phase der Anforderungsdefinition zog sich länger hin als ursprünglich geplant. Die detaillierte Klärung der Anforderungen und die Festlegung der genauen Spezifikationen erwiesen sich als komplexer als erwartet. Eine frühzeitige und präzisere Anforderungsanalyse hätte potenzielle Missverständnisse und spätere Anpassungen vermeiden können.

**Entscheidungsfindung zu Limitationen**: Die Festlegung der Limitationen der Software stellte eine erhebliche Herausforderung dar. Die Entscheidungsfindung war schwierig, da nicht alle möglichen Nutzungsszenarien und technischen Einschränkungen vollständig berücksichtigt wurden. Eine umfassendere Analyse und frühzeitige Diskussion der Limitationen hätten dazu beigetragen, die Software besser zu planen und potenzielle Einschränkungen proaktiv zu adressieren.

**Frühe Integration der Qualitätssicherung**: Ein wichtiger Verbesserungspunkt für zukünftige Projekte ist die frühzeitige Integration der Qualitätssicherung. Anstatt die Qualitätssicherungsmaßnahmen erst nach Abschluss des Großteils des Projekts zu implementieren, sollte diese von Beginn an Bestandteil des Entwicklungsprozesses sein. Die Anwendung von Qualitätssicherungstools und -methoden während der Anforderungsdefinition und des Codings kann dazu beitragen, Probleme frühzeitig zu identifizieren und zu beheben.

**Qualitätssicherung während des gesamten Entwicklungsprozesses**: Die Qualitätssicherung sollte nicht nur am Ende des Projekts, sondern kontinuierlich während des gesamten Entwicklungsprozesses angewendet werden. Dies umfasst regelmäßige Tests, kontinuierliche Code-Reviews und die Implementierung von automatisierten Tests. Eine kontinuierliche Überprüfung der Softwarequalität kann dazu beitragen, die Integrität der Software sicherzustellen und spätere kostspielige Nachbesserungen zu vermeiden.

**Optimierung des Anforderungsmanagements**: Um die Anforderungsdefinition zu verbessern, sollte ein klarer und strukturierter Prozess zur Sammlung und Dokumentation der Anforderungen etabliert werden. Es ist wichtig, alle Stakeholder frühzeitig einzubeziehen und ihre Anforderungen klar zu erfassen. Die Verwendung von Techniken wie User Stories oder Anforderungsworkshops kann helfen, die Anforderungen präziser und effizienter zu definieren.

# **Fazit**

Das Projekt zur Entwicklung des Passwortgenerators stellt einen erfolgreichen Abschluss der festgelegten Ziele dar. Die Software erfüllt die wesentlichen Anforderungen und bietet eine benutzerfreundliche Lösung zur Erstellung sicherer Passwörter. Der Prozess, von der Konzeption bis zur Umsetzung, hat wertvolle Erkenntnisse und Erfahrungen geliefert, die für zukünftige Projekte von Nutzen sein werden.

1. **Erfüllung der Anforderungen**: Der Passwortgenerator wurde gemäß den definierten Anforderungen entwickelt. Die Benutzer können Passwörter in der gewünschten Länge generieren und flexibel verschiedene Zeichenarten wie Großbuchstaben, Kleinbuchstaben, Zahlen und Sonderzeichen auswählen. Dies stellt sicher, dass die Passwörter sowohl sicher als auch anpassbar sind.
2. **Benutzerfreundlichkeit**: Die benutzerfreundliche Oberfläche, die durch HTML und CSS realisiert wurde, ermöglicht eine intuitive Interaktion. Die klare Struktur und einfache Bedienbarkeit tragen dazu bei, dass die Software auch für weniger technisch versierte Benutzer zugänglich ist.
3. **Qualität der Software**: Der Code wurde gründlich getestet und erfüllt die grundlegenden Qualitätsstandards. Die Verwendung von Best Practices im Coding, regelmäßige Code-Reviews und die Einhaltung von Standards haben zur Stabilität und Zuverlässigkeit der Software beigetragen.

Das Projektteam hat wichtige Lektionen aus den Herausforderungen gelernt, insbesondere in Bezug auf die Anforderungsdefinition und die Limitationen der Software. Die längere Dauer der Anforderungsanalyse und die Schwierigkeiten bei der Festlegung der Software-Limitationen haben gezeigt, dass eine detaillierte und frühzeitige Planung unerlässlich ist.

Auf Grundlage der Erfahrungen aus diesem Projekt ergeben sich einige Empfehlungen für zukünftige Entwicklungen:

* **Frühe Integration der Qualitätssicherung**: Qualitätssicherung sollte von Beginn an Teil des Entwicklungsprozesses sein, um Probleme frühzeitig zu identifizieren und zu beheben.
* **Verbesserung des Anforderungsmanagements**: Ein strukturierter Ansatz zur Anforderungsdefinition und -dokumentation kann helfen, Missverständnisse und spätere Anpassungen zu minimieren.
* **Kontinuierliche Überprüfung**: Regelmäßige Tests und Code-Reviews sollten während des gesamten Entwicklungsprozesses durchgeführt werden, um die Softwarequalität zu gewährleisten.

Zusammenfassend zeigt das Projekt, dass ein gut geplantes und umgesetztes Softwareprojekt sowohl in der Funktionalität als auch in der Benutzerfreundlichkeit erfolgreich sein kann. Die gesammelten Erfahrungen bieten wertvolle Einblicke, die zur Verbesserung zukünftiger Projekte beitragen werden. Der Passwortgenerator erfüllt seine Aufgabe effizient und stellt eine zuverlässige Lösung für die sichere Passwortgenerierung dar.