



# SOFTWARE-ENTWICKLUNGSPRAKTIKUM

## OFFICEMANIA SEC0

Software-Entwicklungspraktikum (SEP)  
Sommersemester 2021

### Abnahmetestspezifikation

Auftraggeber  
Technische Universität Braunschweig  
Institut für Systemssicherheit  
Prof. Dr. Konrad Rieck  
Mühlenpförtstraße 23  
38106 Braunschweig

Betreuer: Erwin Quiring, Alexander Warnecke, Jonas Möller

Auftragnehmer:

Name	E-Mail-Adresse
Michael Goslar	m.goslar@tu-bs.de
Paul Hagedorn	paul.hagedorn@tu-bs.de
Johan Kolms	j.kolms@tu-bs.de
Eva Nortmann	e.nortmann@tu-bs.de
Joel Schaub	j.schaub@tu-bs.de
Han Thang	h.thang@tu-bs.de
Fabian Vizi	f.vizi@tu-bs.de
Lukas Wieland	l.wieland@tu-bs.de
Giulia Woywod	g.woywod@tu-bs.de

Braunschweig, 19. Mai 2021

## Bearbeiterübersicht

Kapitel	Autoren	Kommentare
1	Johan Kolms	...
2	Eva Nortmann	...
2.1	Eva Nortmann	...
2.2	Eva Nortmann	...
2.3	Giulia Woywod	...
2.4	Giulia Woywod	...
2.5	Paul Hagedorn	...
3	Johan Kolms	...
3.1	Fabian Vizi, Joel Schaub	...
3.2	Lukas Wieland	...
3.3	Michael Goslar, Johan Kolms	...
4	Han Thang	...

## Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>5</b>
<b>2 Testplan</b>	<b>6</b>
2.1 Zu testende Komponenten . . . . .	6
2.2 Zu testende Funktionen/Merkmale . . . . .	6
2.3 Nicht zu testende Funktionen . . . . .	7
2.4 Vorgehen . . . . .	7
2.5 Testumgebung . . . . .	8
<b>3 Abnahmetest</b>	<b>9</b>
3.1 Zu testende Anforderungen . . . . .	9
3.2 Testverfahren . . . . .	9
3.2.1 Testskripte . . . . .	10
3.3 Testfälle . . . . .	10
3.3.1 Testfall $\langle T100 \rangle$ - Webseitenaufruf erfolgreich . . . . .	11
3.3.2 Testfall $\langle T200 \rangle$ - Karte wird aktualisiert . . . . .	12
3.3.3 Testfall $\langle T300 \rangle$ - Spielerbewegung funktioniert . . . . .	13
3.3.4 Testfall $\langle T400 \rangle$ - Videokonferenz wird gestartet . . . . .	14
3.3.5 Testfall $\langle T500 \rangle$ - Dritte treten der Videokonferenz bei . . . . .	15
3.3.6 Testfall $\langle T600 \rangle$ - Videokonferenz wird beendet . . . . .	16
3.3.7 Testfall $\langle T700 \rangle$ - Die Anzeige für die aktuelle Spielerzahl zeigt alle Spieler an . . . . .	17
3.3.8 Testfall $\langle T800 \rangle$ - Spieler gibt sich einen Namen . . . . .	18
3.3.9 Testfall $\langle T900 \rangle$ - Die richtige Interaktion wird bei Tasten-/Mausklick ausgeführt . . . . .	19
<b>4 Glossar</b>	<b>20</b>

## **Abbildungsverzeichnis**

## 1 Einleitung

Die zu testende Software ist das interaktive Videokonferenztool „OfficeMania“. Um eine hohe Qualität des Endproduktes sicherstellen zu können, werden verschiedene Aspekte während der Entwicklung und nach der Fertigstellung getestet. Je umfangreicher die Tests ausfallen, desto sicherer ist das Erreichen des Ziels, ein Produkt zu erstellen, das eine Benutzung im Alltag einfach und zuverlässig macht.

Diese Testdokumentation orientiert sich am IEEE 829-Standard, einem der bekanntesten Standards für Software-Testdokumentation.

Das Endprodukt umfasst unter anderem folgende zu testende Hauptfunktionen:

- Aufbau einer Verbindung von mehreren Clients zu einem Remote Server
- eine nicht überproportional große Nutzung der Internetbandbreite
- das Bewegen des eigenen Spielers mit der Tastatur
- die Echtzeitdarstellung anderer Spieler
- die dynamische Erstellung und Löschung von Videokonferenzräumen
- die korrekte Darstellung der Benutzeroberfläche im Browser

Der Umfang der zu testenden Neben- und Komfortfunktionen ist erheblich größer und umfasst u.a. gute Video- und Sprachqualität und interaktive Objekte auf der Map.

Die vollständige Erfüllung aller Tests ist von erheblicher Bedeutung für das Gesamtprojekt und dient auch als Mechanismus zum Erkennen, wie gut die Entwicklung der Software voranschreitet.

## 2 Testplan

Im Folgenden wird der Testplan beschrieben. Dabei wird auf die Vorgehensweise und den Umfang der Qualitätssicherung eingegangen, sowie auf die Testgegenstände inklusive ihrer Funktionen. Außerdem werden aufgelistet, welche Maßnahmen für das Testen erforderlich sind und zu welchen Risiken es gegebenenfalls kommen könnte.

### 2.1 Zu testende Komponenten

In diesem Abschnitt werden alle zu testenden Komponenten aufgeführt. Dies bildet also eine Vorlage für den Technischen Entwurf.

- **Komponente  $\langle C10 \rangle$ : Map**

Die Map muss richtig dargestellt werden können, sodass die Büroumgebung verständlich ist. Außerdem muss dabei getestet werden, ob Wände, Türen und andere Objekte als solche funktionieren, sodass die Map wie gedacht verwendet werden kann.

- **Komponente  $\langle C20 \rangle$ : Videochat**

Der Videochat muss darauf getestet werden, ob er sich öffnet, wenn sich Spieler nebeneinander befinden. Außerdem muss der Videochat an sich funktionieren. Das bedeutet, dass getestet werden muss, ob alle Beteiligten sich sehen und hören können.

- **Komponente  $\langle C30 \rangle$ : Player**

Der Spieler muss sich auf der Map bewegen können. Das sollte nicht stockend funktionieren, sondern möglichst flüssig. Davon abgesehen muss man die Spielerbewegungen anderer Nutzer in Echtzeit sehen können. Der Standort muss also richtig an den Server übermittelt werden.

### 2.2 Zu testende Funktionen/Merkmale

Hier werden alle Funktionen aufgeführt, die getestet werden müssen.

#### Anwendungsaufruf $\langle F10 \rangle$

**Kartenzeichnung**  $\langle F20 \rangle$

**Spielerbewegung**  $\langle F30 \rangle$

**Start Videokonferenz**  $\langle F40 \rangle$

**Stop Videokonferenz**  $\langle F50 \rangle$

**Anzeige aktueller Spielerzahl**  $\langle F60 \rangle$

**Namensbestimmung**  $\langle F70 \rangle$

**Spielerinteraktion**  $\langle F80 \rangle$

## 2.3 Nicht zu testende Funktionen

Im Projekt gibt es keine Funktionen, die nicht getestet werden sollten.

Von uns genutzte Bibliotheken, Browser, das Debian-Betriebssystem des Servers und ähnliche nicht von uns programmierte Komponenten sind selbstverständlich nicht Teil der von uns definierten Tests.

## 2.4 Vorgehen

Hier wird das allgemeine Vorgehen zum Testen der zu testenden Funktionen beschrieben, um so eine Zeitabschätzung durchführen zu können. Dabei folgen wir dem Vorgehen des V-Modells, wobei die Tests in die folgenden vier Stufen aufgeteilt werden.

### a) Komponententest

Beim Komponententest werden die in 2.1 angegebenen Komponenten unabhängig voneinander getestet. Für die einzelnen Komponenten werden Jest-Testfälle geschrieben. Dazu werden Black-Box Tests erstellt, welche die Funktionen der Komponenten überprüfen.

### b) Integrationstest

Voraussetzung sind die bereits erfolgreich beendeten Komponententests. Die Komponenten werden nach dem Bottom-Up Prinzip integriert, wobei die Komponenten nacheinander integriert werden. Somit werden Fehler in den Schnittstellen schnell gefunden.

### c) Systemtest

Hier wird aus Nutzersicht das Gesamtsystem getestet. So können alle Fehler gefunden werden, die nur im Gesamtsystem auftreten können. Die Testumgebung ist hierbei möglichst nah an der Umgebung, in der die Software später laufen wird.

### d) Abnahme- und Funktionstest

Nachdem alle vorherigen Teststufen durchlaufen wurden und keine Fehler mehr vorhanden sind, kann das System vom Kunden in der Produktumgebung getestet werden, um die Zufriedenheit und die Abnahme des Kunden sicherzustellen.

## 2.5 Testumgebung

Zum Testen selbst kommt das Framework Jest zum Einsatz.

Tests werden sowohl manuell in der IDE ausgeführt, als auch nach dem Pushen auf das remote Git Repository in der CI/CD Pipeline.

Für den Abnahmetest wird der Test in dem erstellten Docker Image durchgeführt.

Dies sorgt für eine sichere/abgeschottete und reproduzierbare Testumgebung, da das Docker Image klar strukturiert aufgebaut ist und auf verschiedenen Systemen laufen kann.

## 3 Abnahmetest

Die oberste Priorität der Tests ist eine zufriedenstellende Abnahme von OfficeMania durch den Kunden. Dazu zählt eine fehlerfreie Software und die Erfüllung der mit dem Kunden abgesprochenen Punkte aus dem Pflichtenheft.

### 3.1 Zu testende Anforderungen

In diesem Abschnitt werden alle zu testenden Anforderungen aufgeführt

Nr	Anforderung	Testfälle	Kommentar
1	$\langle F10 \rangle$ Anwendungsauftrag	3.3.1	$\langle T100 \rangle$
2	$\langle F20 \rangle$ Kartenzeichnung	3.3.2	$\langle T200 \rangle$
3	$\langle F30 \rangle$ Spielerbewegung	3.3.3	$\langle T300 \rangle$
4	$\langle F40 \rangle$ Start Videokonferenz	3.3.4, 3.3.5	$\langle T400 \rangle, \langle T500 \rangle$
5	$\langle F50 \rangle$ Stop Videokonferenz	3.3.6	$\langle T600 \rangle$
6	$\langle F60 \rangle$ Anzeige aktueller Spielerzahl	3.3.7	$\langle T700 \rangle$
7	$\langle F70 \rangle$ Namensbestimmung	3.3.8	$\langle T800 \rangle$
8	$\langle F80 \rangle$ Spielerinteraktion	3.3.9	$\langle T900 \rangle$

### 3.2 Testverfahren

Im Nachfolgenden wird das Testverfahren kurz erklärt. Dabei werden hauptsächlich Black-Box Tests durchgeführt, welche ohne Wissen über die innere Struktur des Codes auskommen. Somit wird die Funktionalität auf Basis der möglichen Eingabewerte überprüft.

### **3.2.1 Testskripte**

Die meisten Tests werden manuell durchgeführt. Tests, die von Jest Test übernommen werden können, werden möglichst auch automatisiert durchgeführt.

## **3.3 Testfälle**

Hier werden die individuellen Testfälle für die Produktfunktionen aus dem Pflichtenheft ausgeführt. Es wird vor Allem auf die Vor- und Erfolgsbedingungen, sowie auf die Abhängigkeiten zwischen den Testfällen, eingegangen.

### **3.3.1 Testfall $\langle T100 \rangle$ - Webseitenaufruf erfolgreich**

#### **Ziel**

Sicherstellung, dass ein Zugriff auf den Servers korrekt erfolgt.

#### **Objekte/Methoden/Funktionen**

$\langle F10 \rangle$  beschreibt die Ausführung der Webapplikation bei Aufruf der Internetseite.

#### **Pass/Fail Kriterien**

Überprüfung der Ausgabe im Browser. Test ist erfolgreich, wenn der Client erfolgreich OfficeMania beitreten konnte.

#### **Vorbedingung**

Der Server muss online sein.

#### **Einzelschritte**

1. In Suchfeld klicken.
2. Die URL der Webapplikation eingeben.
3. Mit Enter bestätigen.
4. Überprüfen, ob man erfolgreich dem Server beitreten konnte.

#### **Beobachtungen / Log / Umgebung**

Beobachte, ob der Verbindungsaufbau mit dem Server funktioniert hat.

#### **Besonderheiten**

#### **Abhängigkeiten**

### **3.3.2 Testfall $\langle T200 \rangle$ - Karte wird aktualisiert**

#### **Ziel**

Sicherstellung, dass der richtige Ausschnitt der Map berechnet wurde.

#### **Objekte/Methoden/Funktionen**

$\langle F20 \rangle$  beschreibt die Berechnung des richtigen Kartenabschnitts.

#### **Pass/Fail Kriterien**

Überprüfung der Ausgabe im Browser. Test ist erfolgreich, wenn der richtige Ausschnitt der Map berechnet wurde.

#### **Vorbedingung**

Der Server muss online sein und die Webapplikation muss durch eine stabile Internetverbindung erreicht werden können.

#### **Einzelschritte**

1. Als Spieler dem Server beitreten.
2. Spieler bewegen.
3. Überprüfen, ob der richtige Mapausschnitt berechnet wurde.

#### **Beobachtungen / Log / Umgebung**

Beachte, ob der berechnete Mapausschnitt mit dem Erwarteten übereinstimmt.

#### **Besonderheiten**

Das Betreten des Servers und die Weitergabe der neuen Position nach einer Bewegung muss erfolgreich ablaufen.

#### **Abhängigkeiten**

Von Testfall  $\langle T100 \rangle$  und  $\langle T300 \rangle$  abhängig.

### **3.3.3 Testfall $\langle T300 \rangle$ - Spielerbewegung funktioniert**

#### **Ziel**

Überprüfung ob der Spieler sich bei Userinput in die richtige Richtung bewegt.

#### **Objekte/Methoden/Funktionen**

$\langle F30 \rangle$  beschreibt die Bewegung des Spielers vom Nutzer.

#### **Pass/Fail Kriterien**

Der Test ist erfolgreich, sollte nach der Simulation eines Tastendrucks die Position in x bzw. y um 1 erhöht bzw. verringert worden sein.

Der Test ist fehlgeschlagen, sollte sich nach der Simulation eines Tastendrucks die Position in x bzw. y gar nicht geändert, zu viel geändert, oder in die falsche Richtung geändert haben.

#### **Vorbedingung**

Der Server muss online sein und die Webapplikation muss durch eine stabile Interverbindung erreicht werden können.

#### **Einzelschritte**

1. Als Spieler einloggen.
2. Spieler bewegen.
3. Überprüfe, ob der Spieler sich richtig bewegt.

#### **Beobachtungen / Log / Umgebung**

Beachte, ob der Spieler in die richtige Richtung gelaufen ist und die richtige Distanz zurückgelegt hat.

#### **Besonderheiten**

-

#### **Abhängigkeiten**

Von Testfall  $\langle T100 \rangle$  abhängig.

### **3.3.4 Testfall $\langle T400 \rangle$ - Videokonferenz wird gestartet**

#### **Ziel**

Überprüfung, ob eine Videokonferenz korrekt gestartet wird, wenn sich zwei Spieler einander nähren.

#### **Objekte/Methoden/Funktionen**

$\langle F40 \rangle$  beschreibt die Annäherung von Spielern.

#### **Pass/Fail Kriterien**

Überprüfung des Starts der Videokonferenz durch zwei Spieler, die sich einander nähren.  
Test erfolgreich, wenn die Videokonferenz startet, wenn zwei Spieler nah aneinander stehen.  
Test nicht erfolgreich, wenn die Videokonferenz nicht startet oder ein dritter Spieler auch beitritt, der aber nicht in der Nähe ist.

#### **Vorbedingung**

Auf dem Server sind mindestens 3 Spieler, davon zwei für die Videokonferenz und ein Spieler als Referenz, der nicht in der Nähe steht. Der dritte Spieler darf nicht der Videokonferenz beitreten.

#### **Einzelschritte**

1. Ein Spieler steht auf der Map.
2. Ein zweiter Spieler nährt sich an den ersten an.

Ziel: Sind die Spieler nah genug aneinander, startet eine Videokonferenz mit genau diesen beiden Spielern als einzige Teilnehmer.

#### **Beobachtungen / Log / Umgebung**

Beobachtet wird, ob die Erkennung funktioniert, dass sich zwei Spieler aneinander nähren und die Videokonferenz startet.

#### **Besonderheiten**

-

#### **Abhängigkeiten**

Von Testfall  $\langle T100 \rangle$  und  $\langle T300 \rangle$  abhängig.

### **3.3.5 Testfall $\langle T500 \rangle$ - Dritte treten der Videokonferenz bei**

#### **Ziel**

Überprüfung, ob eine Videokonferenz korrekt gestartet wird, wenn sich mehr als zwei Spieler einander nähren.

#### **Objekte/Methoden/Funktionen**

$\langle F40 \rangle$  beschreibt die Annäherung von Spielern.

#### **Pass/Fail Kriterien**

Überprüfung des Starts der Videokonferenz durch mehrere Spieler, die sich einander nähren.

Test erfolgreich, wenn die Videokonferenz startet, wenn die Spieler nah aneinander stehen und auch dann weiterhin besteht, wenn der oder die Spieler, die zuerst in der Konferenz waren, weggehen.

Test nicht erfolgreich, wenn die Videokonferenz nicht startet oder abbricht, wenn der Spieler weggeht, der zuerst in der Videokonferenz war.

#### **Vorbedingung**

Auf dem Server sind mindestens 4 Spieler, davon mindestens drei für die Videokonferenz und ein Spieler als Referenz, der nicht in der Nähe steht. Der einzeln stehende Spieler darf nicht der Videokonferenz beitreten.

#### **Einzelschritte**

1. Ein Spieler steht auf der Map.
2. Mehr als ein weiterer Spieler nähren sich dem ersten Spieler.
3. Der erste Spieler entfernt sich.

Ziel: Sind die Spieler nah genug aneinander, startet eine Videokonferenz mit genau diesen Spielern als einzige Teilnehmer. Entfernt sich der erste Spieler, verlässt er die Videokonferenz, aber für alle anderen bleibt sie bestehen.

#### **Beobachtungen / Log / Umgebung**

Beobachtet wird, ob die Erkennung funktioniert, dass sich mehrere Spieler aneinander nähren und die Videokonferenz startet und auch dann weiterhin besteht, falls der erste Spieler die Gruppe verlässt.

#### **Besonderheiten**

-

#### **Abhängigkeiten**

Von Testfall  $\langle T100 \rangle$  und  $\langle T300 \rangle$  abhängig.

### **3.3.6 Testfall $\langle T600 \rangle$ - Videokonferenz wird beendet**

#### **Ziel**

Überprüfung, ob eine Videokonferenz korrekt beendet wird, wenn sich zwei oder mehr Spieler voneinander trennen.

#### **Objekte/Methoden/Funktionen**

$\langle F50 \rangle$  beschreibt das Weglaufen von Spielern voneinander.

#### **Pass/Fail Kriterien**

Überprüfung des Beendens der Videokonferenz durch mehrere Spieler, die sich von einander entfernen.

Test erfolgreich, wenn die Videokonferenz endet, wenn die Spieler voneinander weglaufen.

Test nicht erfolgreich, wenn die Videokonferenz bestehen bleibt, wenn die Spieler voneinander weglaufen.

#### **Vorbedingung**

Auf dem Server sind mindestens 4 Spieler, davon mindestens drei für die Videokonferenz und ein Spieler als Referenz, der nicht in der Nähe steht. Der einzeln stehende Spieler darf nicht der Videokonferenz beitreten sein.

#### **Einzelschritte**

1. Mehrere Spieler stehen beieinander und sind in einer Videokonferenz.
2. Ein oder mehrere Spieler entfernen sich.
3. Kein Spieler steht mehr in der Nähe eines anderen.

Ziel: Die Spieler verlassen die Videokonferenz, sobald sie sich entfernen. Kein Spieler ist mehr in einer Videokonferenz, wenn er nicht in der Nähe eines anderen ist.

#### **Beobachtungen / Log / Umgebung**

Beobachtet wird, ob die Erkennung funktioniert, dass sich mehrere Spieler voneinander entfernen und die Videokonferenz dadurch endet.

#### **Besonderheiten**

#### **Abhängigkeiten**

Von Testfall  $\langle T100 \rangle$ ,  $\langle T300 \rangle$  und  $\langle T400 \rangle$  abhängig.

### **3.3.7 Testfall $\langle T700 \rangle$ - Die Anzeige für die aktuelle Spielerzahl zeigt alle Spieler an**

#### **Ziel**

Überprüfung, ob alle Namen der beigetretenden Spieler angezeigt werden.

#### **Objekte/Methoden/Funktionen**

$\langle F60 \rangle$  beschreibt die Anzeige der aktuell beigetretenden Spieler.

#### **Pass/Fail Kriterien**

Überprüfung der Ausgabe durch Testperson. Test ist erfolgreich, wenn alle Namen der beigetretenden Spieler richtig angezeigt werden.

#### **Vorbedingung**

Der Server muss online sein und die Webapplikation muss durch eine stabile Interverbinding erreicht werden können. Dazu müssen mindestens 2 Personen dem Server beigetreten sein als Spieler.

#### **Einzelschritte**

1. Spieler betreten den Server.
2. Spieler geben sich Namen.
3. Überprüfe ob alle Namen richtig angezeigt werden.

#### **Beobachtungen / Log / Umgebung**

Beobachte, ob die Namen richtig angezeigt werden.

#### **Besonderheiten**

Das Betreten des Servers und das Setzen des Namens muss erfolgreich ablaufen.

#### **Abhängigkeiten**

Von Testfall  $\langle T100 \rangle$  und  $\langle T800 \rangle$  abhängig.

### **3.3.8 Testfall $\langle T800 \rangle$ - Spieler gibt sich einen Namen**

#### **Ziel**

Überprüfung, ob eine Namensänderung erfolgreich war.

#### **Objekte/Methoden/Funktionen**

$\langle F70 \rangle$  beschreibt die Änderung des Namens von einem Spieler.

#### **Pass/Fail Kriterien**

Überprüfung der Ausgabe durch Testperson. Test ist erfolgreich, wenn der neue Name den alten Namen ersetzt hat.

#### **Vorbedingung**

Der Server muss online sein und die Webapplikation muss durch eine stabile Interverbinding erreicht werden können.

#### **Einzelschritte**

1. Als Spieler dem Server beitreten.
2. Menü öffnen.
3. Namen ändern.
4. Überprüfe ob der neue Name übernommen wurde.

#### **Beobachtungen / Log / Umgebung**

Beobachte, ob der Spieler seinen Namen geändert hat.

#### **Besonderheiten**

Das Betreten des Servers und Öffnen des Menüs mus erfolgreich ablaufen.

#### **Abhängigkeiten**

Von Testfall  $\langle T100 \rangle$  abhängig.

### **3.3.9 Testfall $\langle T900 \rangle$ - Die richtige Interaktion wird bei Tasten-/Mausklick ausgeführt**

#### **Ziel**

Überprüfung, ob die gewünschte Aktion bei der Interaktion mit einem Objekt ausgeführt wird.

#### **Objekte/Methoden/Funktionen**

$\langle F80 \rangle$  beschreibt die Interaktion mit Objekten.

#### **Pass/Fail Kriterien**

Überprüfung der Ausgabe durch Testperson. Test ist erfolgreich, wenn die gewünschte Aktion, bei der Interaktion ausgeführt wird.

#### **Vorbedingung**

Der Server muss online sein und die Webapplikation muss durch eine stabile Interverbindung erreicht werden können.

#### **Einzelschritte**

1. Als Spieler dem Server beitreten.
2. Spieler zu dem zu testenden Objekt bewegen.
3. Vor dem Objekt die Interaktionstaste drücken.
4. Überprüfe ob die richtige Aktion ausgeführt wurde.

#### **Beobachtungen / Log / Umgebung**

Beobachte, ob die richtige Aktion ausgeführt wird.

#### **Besonderheiten**

Das Betreten des Servers und die Bewegung muss erfolgreich ablaufen.

#### **Abhängigkeiten**

Von Testfall  $\langle T100 \rangle$ ,  $\langle T200 \rangle$  und  $T\langle 300 \rangle$  abhängig.

## 4 Glossar

**CI/CD Pipeline:** Eine Vorgehensweise zur Verbesserung der Softwarebereitstellung, wie z.B. durch den Einsatz von DevOps.

**Client:** Clients sind Programme oder Computer, die auf Server zugreifen können und von denen Daten oder Dienste abrufen.

**DevOps:** DevOps sorgt dafür, dass Kunden durchgehend hochwertige Produkte erhalten, indem es Menschen, Prozesse und Technologien vereint.

**Debian:** Debian ist ein Betriebssystem.

**Black-Box Test:** Der Black-Box Test ist ein Testverfahren, wo das System von außen beurteilt wird, wobei der Nutzer nicht weiß, was innerhalb des Systems passiert.

**Git:** Git ist ein Open-Source-Tool zur verteilten Versionskontrolle von Software. Jedes Teammitglied hat jederzeit Zugriff auf die aktuellste Version vom Code.

**IDE:** IDE (Integrated Development Environment) ist eine Software, welche gängige Entwickler-Tools vereint, um Anwendungen zu programmieren.

**Jest:** Jest ist ein JavaScript-Testframework.

**Remote Server:** Remote Server bieten Zugriff auf gemeinsam genutzte Daten und Objekte.