Software-Entwicklungspraktikum

OfficeMania SEC0

Software-Entwicklungspraktikum (SEP)
Sommersemester 2021

Testspezifikation

Auftraggeber
Technische Universität Braunschweig
Institut für Systemsicherheit
Prof. Dr. Konrad Rieck
Mühlenpfordtstraße 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, 14. Juli 2021

Bearbeiterübersicht

Kapitel	Autoren	Kommentare
1	Eva, Joel, Han, Johan	...
2	Eva, Joel, Han	...
2.1	Eva, Joel, Han	...
2.2	Eva, Joel, Han	...
2.3	Eva, Joel, Han	...
2.4	Eva, Joel, Han	...
2.5	Eva, Joel, Han, Paul	...
3	Giulia, Michael, Fabian	...
3.1	Giulia, Michael	...
3.2	Giulia, Michael	...
3.3	Giulia, Michael, Johan	...
4	Eva, Joel, Han	...
4.1	Eva, Joel, Han	...
4.2	Eva, Joel, Han	...
4.3	Eva, Joel, Han	...
5	Lukas	...
5.1	Lukas	...
5.2	Lukas	...
5.3	Lukas, Michael	...
6	Fabian	...

Inhaltsverzeichnis

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 Hauptfuktionen:
- 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 wird 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.

- **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 $\langle C20 \rangle$: 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. Ebenfalls sollte das Aussehen des Charakter mithilfe eines Charakter Auswahlmenü veränderbar sein.

- **Komponente $\langle C30 \rangle$: Videochat**
  Der Videochat muss darauf getestet werden, ob er sich öffnet, wenn sich Spieler nebeneinander befinden. Ebenfalls ist darauf zu achten, dass der Videochat sich auch dann nur öffnet, wenn sich die Spieler im selben Raum befinden. Außerdem muss der Videochat an sich funktionieren. Das bedeutet, dass getestet werden muss, ob alle Beteiligten sich sehen und hören können.

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 $\langle F80 \rangle$

Charakterauswahl $\langle F90 \rangle$

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	$\langle F10 \rangle$ Anwendungsaufruf	<T100>	Sicherstellung eines funktionierenden Serverzugriffs
2	$\langle F20 \rangle$ Kartenzeichnung	<T200>	Überprüfung der Kartenzeichnung
3	$\langle F30 \rangle$ Spielerbewegung	<T300>	Test für richtige Bewegung eines Spielers
4	$\langle F40 \rangle$ Start Videokonferenz	<T400>, <T500>	Überprüfung des Videokonferenzstarts, wenn Spieler sich nähern
5	$\langle F50 \rangle$ Stop Videokonferenz	<T600>	Test zur richtigen Beendung einer Videokonferenz, wenn sich Spieler entfernen
6	$\langle F60 \rangle$ Anzeige aller aktueller Spielernamen	<T700>	Test, ob alle Spielernamen der Spieler auf dem Server angeziegt werden
7	$\langle F70 \rangle$ Namensbestimmung	<T800>	Validierung, dass Namensänderungen korrekt funktioneren

8	⟨$F80$⟩ Spielerinteraktion	<T900>	Überprüfung, ob bei Interaktion mit Objekten die richtigen Aktionen ausgeführt werden
9	⟨$F90$⟩ Charakterauswahl	<T1000>	Validierung, dass die Charakterauswahl korrekt funktionert

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

<F10> 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. Test ist fehlgeschlagen, wenn die Seite nicht aufgerufen werden 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

<F20> beschreibt die Berechnung des richtigen Kartenabschnitts.

Pass/Fail Kriterien

Überprüfung der Ausgabe im Browser. Test ist erfolgreich, wenn der richtige Auschnitt der Map berechnet wurde. Test ist fehlgeschlagen, wenn die Map garnicht, oder ein falscher Kartenauschnitt gezeichnet wurde.

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 bewegen.
3. Überprüfen, ob der richtige Mapausschnitt berechnet wurde.

Beobachtungen / Log / Umgebung

Beobachte, 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 <T100> und <T300> 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 Richtung 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 Internetverbindung erreicht werden können.

Einzelschritte

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

Beobachtungen / Log / Umgebung

Beobachte, ob der Spieler in die richtige Richtung gelaufen ist und richtige Distanz (24 Pixel pro Schritt) zurückgelegt hat. Hierbei ist zu beachten, dass sich die Map in die entgegengesetzte Richtung bewegt, wie der Spieler sich bewegen möchte und dass der Charakter in die Richtung schaut in die er sich bewegt.

Besonderheiten

-

Abhängigkeiten

Von Testfall <T100> 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

<F40> 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.
3. Eine Videokonferenz startet mit 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 <T100> und <T300> 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. 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.

Objekte/Methoden/Funktionen

<F40> beschreibt die Annährung 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.
4. Dieser verlässt Konferenz, für andere 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 <T100> und <T300> 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 entfernen.

Objekte/Methoden/Funktionen

<F50> 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.
4. Kein Spieler ist mehr in einer Videokonferenz.

Beobachtungen / Log / Umgebung

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

Besonderheiten

-

Abhängigkeiten

Von Testfall <T100>, <T300> und <T400> abhängig.

3.3.7 Testfall $\langle T700 \rangle$ - Die Anzeige für die aktuellen Spielernamen zeigt alle Spieler an

Ziel

Überprüfung, ob alle Namen der beigetretenden Spieler angezeigt werden.

Objekte/Methoden/Funktionen

<F60> 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. Test ist fehlgeschlagen, wenn nicht alle, oder falsche Spielernamen angezeigt werden.

Vorbedingung

Der Server muss online sein und die Webapplikation muss durch eine stabile Internetverbindung 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 <T100> und <T800> abhängig.

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

Ziel

Überprüfung, ob eine Namensänderung erfolgreich war.

Objekte/Methoden/Funktionen

<F70> beschreibt die Änderung des Namens von einem Spieler.

Pass/Fail Kriterien

Überprüfung der Ausgabe durch zwei oder mehrere Testpersonen. Test ist erfolgreich, wenn der neue Name den alten Namen ersetzt hat. Test ist fehlgeschlagen, wenn der alte Name nicht ersetzt wurde, oder durch einen falschen Namen ersetzt 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. Einstellungen öffnen.
3. Namen ändern.
4. Überprüfe ob der neue Name übernommen wurde.
5. Mindestens ein weiterer Spieler überprüft, ob der neue Name übernommen wurde.

Beobachtungen / Log / Umgebung

Beobachte, ob der Spieler seinen Namen geändert hat.

Besonderheiten

Das Betreten des Servers und Öffnen der Einstellungen muss bei allen beteiligten Spielern erfolgreich ablaufen.

Abhängigkeiten

Von Testfall <T100> 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

<F80> 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. Test ist fehlgechlagen, wenn die falsche AKtion, oder garkeine Aktion ausgeführt 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 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 <T100>, <T200> und T<300> abhängig.

3.3.10 Testfall $\langle T1000 \rangle$ - Spieler ändert das Aussehen seines Charakters

Ziel

Überprüfung, ob eine Charakterauswahl erfolgreich war.

Objekte/Methoden/Funktionen

<F90> beschreibt die Änderung des Aussehens von einem Spieler.

Pass/Fail Kriterien

Überprüfung der Ausgabe durch zwei oder mehrere Testpersonen. Test ist erfolgreich, wenn das neue Erscheinungsbild des Charakters das alte Erscheinungsbild ersetzt hat. Test ist fehlgeschlagen, wenn das alte Erscheinungsbild nicht ersetzt wurde, oder durch ein falsches Erscheinungsbild ersetzt 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. Einstellung öffnen.
3. Charakter ändern.
4. Überprüfe ob der neue Charakter übernommen wurde.
5. Mindestens ein weiterer Spieler überprüft, ob der neue Charakter übernommen wurde.

Beobachtungen / Log / Umgebung

Beobachte, ob der Spieler seinen Charakter geändert hat.

Besonderheiten

Das Betreten des Servers und Öffnen des Menüs muss bei allen beteiligten Spielern erfolgreich ablaufen.

Abhängigkeiten

Von Testfall <T100> abhängig.

4 Integrationstest

In diesem Kapitel behandeln wir spezifisch die Vorgehensweisen beim Integrationstest. Hierbei wird beim Integrationstest die korrekte Zusammenarbeit der Komponenten überprüft und sichergestellt.

4.1 Zu testende Komponenten

Nr	Komponenten	Testfälle	Kommentar
1	$\langle C10 \rangle$ Map	<T1100>, <T1200>	
2	$\langle C20 \rangle$ Player	<T1200>, <T1300>	
3	$\langle C30 \rangle$ Videochat	<T1100>, <T1300>	

4.2 Testverfahren

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

4.2.1 Testskripte

Wir nutzen keine Testskripte, da wir manuell testen, um besser feststellen zu können, ob auf der Map alles funktioniert.

4.3 Testfälle

In diesem Abschnitt werden die Testfälle zwischen den Komponenten genauer beschrieben.

4.3.1 Testfall $\langle T1100 \rangle$ - Komponente C10 + C20

Ziel

Hier testen wir, ob die Koordinaten richtig aktualisiert werden.

Objekte/Methoden/Funktionen

F20, F30, updatePosition(player: Player, room: Room),
updateOwnPosition(player: Player, room: Room, collisionInfo: solidInfo[][]),
convertMapData(mapdata: string, room: Room, canvas: HTMLCanvasElement)

Pass/Fail Kriterien

Überprüfung der Ausgabe durch zwei oder mehrere Testpersonen. Test ist erfolgreich, wenn die Koordinaten richtig aktualisiert werden, also wenn man sich beim Gehen in die geplante Richtung bewegt und auch alle anderen Spieler sehen können, wohin man sich bewegt. Test ist fehlgeschlagen, wenn die Koordinaten falsch aktualisiert werden und man sich somit nicht richtig bewegt.

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. Bewegen.

Beobachtungen / Log / Umgebung

Beobachte, ob der Spieler sich seiner Eingabe entsprechend richtig bewegt.

Besonderheiten

Das Betreten des Servers muss erfolgreich ablaufen.

Abhängigkeiten

Von Testfall <T100> und <T300> abhängig.

4.3.2 Testfall $\langle T1200 \rangle$ - Komponente C10 + C30

Ziel

Hier testen wir die Abstandsprüfung zwischen zwei oder mehreren Playern.

Objekte/Methoden/Funktionen

F40, F50, nearbyPlayerCheck(players: PlayerRecord, ourPlayer: Player)

Pass/Fail Kriterien

Überprüfung der Ausgabe durch Testperson. Test ist erfolgreich, wenn der Abstand richtig bestimmt wurde. Test ist fehlgeschlagen, wenn der Abstand fehlerhaft bestimmt wurde. Andere Spieler dürfen maximal 223,6 Pixel, also ca. 9 Blöcke von einem entfernt sein damit sie in Reichweite sind.

Vorbedingung

Der Server muss online sein, die Webapplikation muss durch eine stabile Internetverbindung erreicht werden können und es müssen mindestens zwei Spieler online sein.

Einzelschritte

1. Als Spieler dem Server beitreten.
2. Warten, bis ein anderer Spielern beitritt.
3. Gehe nah zu dem anderen Spielern.
4. Von anderen Spielern distanzieren.

Beobachtungen / Log / Umgebung

Beobachte, ob der Abstand immer richtig berechnet wird.

Besonderheiten

Das Betreten des Servers muss bei allen Spielern erfolgreich ablaufen.

Abhängigkeiten

Von Testfall <T100> abhängig.

4.3.3 Testfall $\langle T1300 \rangle$ - Komponente C20 + C30

Ziel

Hier testen wir, ob richtig erkannt wird, ob sich Spieler in der Nähe befinden.

Objekte/Methoden/Funktionen

F40, F50, nearbyPlayerCheck(players: PlayerRecord, ourPlayer: Player)

Pass/Fail Kriterien

Überprüfung der Ausgabe durch Testperson. Test ist erfolgreich, wenn richtig erkannt wird, ob sich andere Spieler im gleichen Raum und in der gewünschten Reichweite befinden, damit der Videochat funktioniert. Der Test ist fehlgeschlagen, wenn Spieler als in der Nähe erkannt werden, obwohl sie sich im anderen Raum befinden oder sich nicht in der Nähe befinden. Außerdem schlägt der Test fehl, wenn Spieler als nicht in der Nähe erkannt werden, obwohl sie sich in der Nähe und im gleichen Raum befinden.

Vorbedingung

Das Betreten des Servers muss bei allen Spielern erfolgreich ablaufen.

Einzelschritte

1. Als Spieler dem Server beitreten.
2. Warten, bis ein anderer Spielern beitritt.
3. Gehe nah zu den anderen Spielern.
4. Von anderen Spielern distanzieren.

Beobachtungen / Log / Umgebung

Beobachte, ob richtig angezeigt wird, ob sich Spieler in der Nähe befinden oder nicht.

Besonderheiten

Das Betreten des Servers muss bei allen Spielern erfolgreich ablaufen.

Abhängigkeiten

Von Testfall <T100> und <T300> abhängig.

5 Unit-Tests

Dieses Kapitel befasst sich mit den Unit-Tests der Map, Player und Videochat Komponente. Alle relevanten Funktionen werden hierfür einzeln getestet. Somit werden potenzielle Bugs und Implementierungsfehler isoliert und eingegrenzt.

5.1 Zu testende Komponenten

In der folgenden Tabelle werden die Komponenten und ihre zugeordneten Testfälle aufgelistet.

Nr	Komponenten	Testfälle	Kommentar
1	<C10> Map	<T1400>	
2	<C20> Player	<T300>, <T1200>,	
3	<C30> Videochat	<T400>, <T500>, <T600>, <T1300>, <T1500>, <T1600>	

5.2 Testverfahren

Für das Testverfahren werden hauptsächlich Black-Box Tests genutzt. Hierbei wird der Erfolgsfaktor einer Funktion anhand einer Ausgabe, folgend einer bestimmten Eingabe, gemessen. Für den zu beachtenden Erfolgsfaktor wurden Kriterien, sowie eventuelle Beschreibungen, vorbestimmt.

5.2.1 Testskripte

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

5.3 Testfälle

Hier folgen die Testfälle. Zuerst wird das Testziel festgelegt, welches die Bestimmung der relevanten Objekte, Methoden und Funktionen ermöglicht. Die Pass- und Fail-Kriterien werden genau definiert. Es folgen die Vorbedingung, eine Detaillierung der einzelnen Testschritte, und letztendlich mögliche Beobachtungen, Besonderheiten und Abhängigkeiten.

5.3.1 Testfall $\langle T1400 \rangle$ - Mapdaten füllen

Ziel

Testen, ob alle notwendigen Infos an den richtigen stellen der Map gesetzt wurden (Kollision, Interaktionen, RaumID).

Objekte/Methoden/Funktionen

- mapInfo
- solidInfo
- fillSolidInfos(mapInfo): solidInfo[][]

Pass/Fail Kriterien

Überprüfung der Ausgabe durch Testperson. Test ist erfolgreich, wenn die gegeben Daten über einzelne Felder an den richtigen stellen gesetzt wurden. Test schlägt fehl, wenn Daten an falschen stellen gesetzt wurden, fehlen, oder zu viel sind.

Vorbedingung

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

Einzelschritte

1. Als Spieler den Server betreten.
2. Die Map ablaufen.

Beobachtungen / Log / Umgebung

Beobachte für Kollision, RoomID und Interaktionen einzeln, ob die Texturen an den jeweiligen stellen mit den Texturen des jeweiligen Layer im Mapeditor übereinstimmen.

Besonderheiten

Das Betreten des Servers muss erfolgreich ablaufen.

Abhängigkeiten

Von Testfall <T100>, <T200> und <T300> abhängig.

5.3.2 Testfall $\langle T1500 \rangle$ - Audio, Video an-/ ausschalten

Ziel

Auf erfolgreiches An- sowie Ausschalten einer Audio- und Videoübertragung testen.

Objekte/Methoden/Funktionen

toggleMute(string):void, toggleMuteByType(string), toggleCamAudio():void, toggleCamVideo(): void

Pass/Fail Kriterien

Überprüfung der Ausgabe durch Testpersonen. Der Test ist erfolgreich, wenn das Ein- und Ausschalten der Audio- und Videoübertragung wie geplant funktioniert. Der Test ist unerfolgreich, wenn der oben beschriebene Ablauf nicht problemlos funktioniert.

Vorbedingung

Der Server muss online sein, die Webapplikation muss durch eine stabile Internetverbindung erreicht werden können. Mindestens zwei Spieler nähern sich auf der Karte auf die nötige Distanz um die Video-/Audioübertragung zu ermöglichen.

Einzelschritte

1. Zwei oder mehr Spieler nähern sich auf der Karte auf die nötige Distanz um eine Video-/Audioübertragung zu ermöglichen
2. Die Video-/Audiofunktion wird von Spieler 1 gestartet
3. Die Veränderung wird von Spieler 2 beobachtet
4. Die Video-/Audiofunktion wird von Spieler 1 beendet
5. Die Veränderung wird von Spieler 2 beobachtet

Beobachtungen / Log / Umgebung

Ein Spieler muss die Veränderung der Video- und Audioübertragung beobachten.

Besonderheiten

Der Test muss für die Audio- und die Videofunktion getestet werden.

Abhängigkeiten

Von Testfall <T100>, <T300>, <T400>, <T500>, <T600> und <T1300>

5.3.3 Testfall $\langle T1600 \rangle$ - Bildschirmfreigabe

Ziel

Überprüfung, ob die Bildschirmübertragung richtig funktioniert.

Objekte/Methoden/Funktionen

toggleMute(string): void, toggleSharing((boolean) => void): void

Pass/Fail Kriterien

Überprüfung der Ausgabe durch Testperson. Test erfolgreich, wenn der ausgewählte Bildschirm korrekt für sich und andere Angezeigt wird. Test schlägt fehl, wenn kein Bildschirm, oder der falsche Bildschirm übertragen wird.

Vorbedingung

Der Server muss online sein, die Webapplikation muss durch eine stabile Internetverbindung erreicht werden können. Es müssen mindestens zwei Personen auf dem Server sein.

Einzelschritte

1. Zwei Spieler betreten den Server.
2. Spieler bewegen sich aufeinander zu, sodass ein Videochat gestartet wird.
3. Ein Spieler startet eine Bildschirmübertragung.

Beobachtungen / Log / Umgebung

Beide Spieler gucken, ob der richtige Bildschirm übertragen wird.

Besonderheiten

keine

Abhängigkeiten

Von Testfall <T100>, <T300>, <T400> und <T1300>

6 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 das 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 System 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.