Software Requirements Specification

Inhaltsverzeichnis

1 Vorwort

1.1 Zielpublikum dieses Dokuments

Dieses Dokument umfasst die technischen Anforderungen an die Applikation *despresso*. Es dient den Kunden für die Überprüfung der Anforderungen an ihr Produkt, der Projektleitung um die Kosten des Projekts abzuschätzen, den System Engineers um die spezifischen Systemkomponenten zu planen, den Test Engineers um die Validation Tests zu entwickeln sowie den Maintainance Engineers um die Beziehung zwischen den Systemkomponenten zu verstehen.

1.2 Änderungsverzeichnis

Datum	Änderung	Autor	Dokumentname
01.04.2019	Dokument erstellt	Lukas Zoller	Software_Requirements_Specification_V1
04.04.2019	Kapitel eingefügt	Lukas Zoller	Software_Requirements_Specification_V2

2 Einleitung

Die in diesem Dokument beschriebene Applikation *despresso* soll im Problemfeld von Depressionspatienten Unterstützung bieten. Depressionspatienten sind mit zwei Hauptproblemen konfrontiert: Antriebslosigkeit und Selbstzweifel. Antriebslosigkeit resultiert bei Depressionspatienten oft in einer fehlenden Tagestruktur, Ideenlosigkeit wie man die Zeit gestalten könnte und einer tiefen Motivation für Aktivitäten, sei es für notwendige Aktivitäten wie Putzen, Duschen, etc. oder Freizeitaktivitäten wie z.B. Spazieren. Selbstzweifel führen bei Depressionspatienten zu einem Gefühlsleben, das häufig geprägt ist von einer abwärts gerichteten Depressionsspirale. Dabei resultiert das Fehlen positiver Erlebnisse und Gefühle zu Unlust und Inaktivität und dies wiederum zu einer noch schlechteren Stimmung.

Die Applikation *despresso* soll Unterstützung bei diesen Problemen bieten. Drei Hauptfunktionen werden in dieser Applikation realisiert. Erstens können positive Gefühle mit verschiedenem Detaillierungsgrad dokumentiert werden. Dies ermöglicht dem Patienten, Rückblicke auf seine Gefühlslage zu werfen und so zu erkennen, dass seine Gefühlswelt nicht nur aus negativen Gefühlen besteht, sondern auch positive Emotionen vorhanden sind. Das ist essentiell für das Durchbrechen oder Abschwächen der Depressionsspirale. Diese Dokumentation kann vom Therapeuten konsultiert werden um die Verfassung des Patienten zu überwachen und um die Behandlung anzupassen. Eine zweite Hauptfunktion bietet der Kalender. Darin kann allein oder mit dem Therapeuten eine Tages- und Wochenstruktur definiert werden. Die täglichen Aufgaben wie Duschen, Kochen, etc. können nach erfolgreichem Bewältigen abgehakt werden. Die dritte Hauptfunktion von *despresso* besteht im Vorschlagen von Aktivitäten. So werden dem Patienten einerseits Aktivitäten gemäss seiner Gefühlslage vorgeschlagen, falls ein Eintrag eines negativen Gefühls durch den Patienten erfolgt. Andererseits kann der Patient selbstständig Aktivitätsvorschläge fordern und diese durch gezielte Filtereinstellungen (Gefühlslage, Örtlichkeit, etc.) einschränken. Diese Aktivitätsvorschläge sollen dem Patienten helfen, unerwünscht inaktive Phasen zu verringern.

3 Glossar

Das Glossar umfasst die Erklärung der technischen Begriffe, die in diesem Dokument verwendet werden.

Backend

Für einen eingeschränkten Benutzerkreis (Therapeut) zugängliche Bereich der Applikation.

Cloud

Infrastruktur, die über einen Internetzugang verfügbar ist (z.B. Die Datenbank auf welcher die Kalendereinträge von *despresso* abgespeichert werden liegt in der Cloud).

Depressionsspirale

Eine sich verstärkender Zirkel aus negativen Gefühlen, Antriebslosigkeit und daraus resultierenden negativen Gefühlen.

Frontend

Der Teil der Applikation, welcher der Nutzer sehen kann.

Java

Java ist eine objektorientierte Programmiersprache.

Unit Tests

Unit Tests prüfen einzelne Funktionen einer Applikation.

Use Cases

Use Cases beschreiben prototypische Handlungen mit einer Applikation (z.B. Kalendereintrag erstellen).

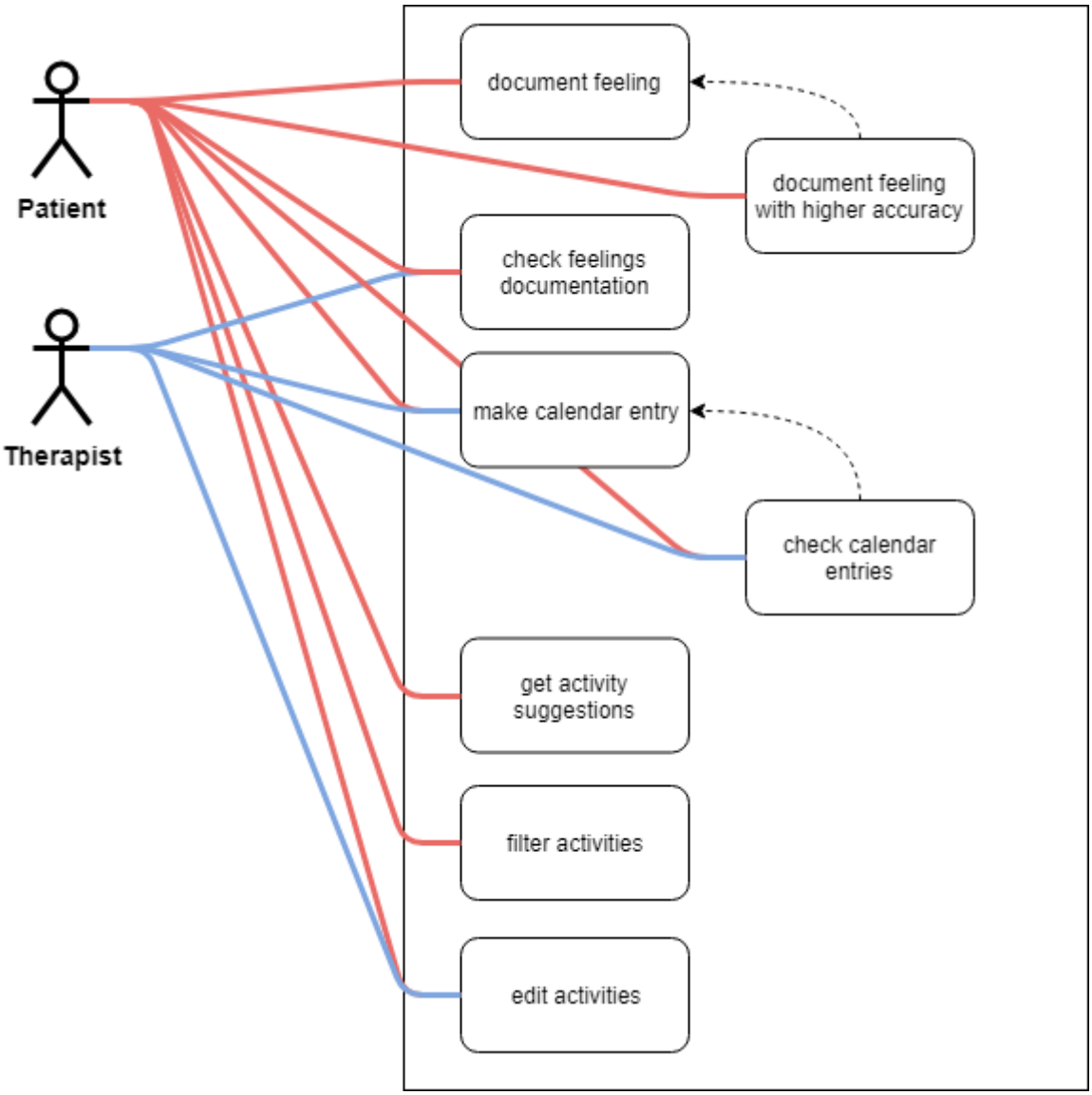
Vaadin

Vaadin ist ein open Source Webframework für die Programmierung von unter Anderem Client-Server-Applikationen mit Java.

4 User Requirements Definition

In diesem Kapitel werden die Funktionen der Applikation *despresso* beschrieben. Im ersten Unterkapitel werden in einem Diagramm die verschiedenen Use Cases sowie ihre Beziehung zueinander dargestellt. Im folgenden Unterkapitel werden drei Use Cases auf Handlungsebene beschrieben.

4.1 Use Case Diagram



4.2 Use Cases

4.2.1 Use Case 1: Termin erstellen

Nr. und Name:	1: Termin im Kalender erstellen
Szenario:	Benutzer befindet sich in einem leeren Terminformular
Kurzbeschreibung:	Der Benutzer möchte (z.B. zusammen mit seinem Therapeuten) einen Termin erstellen, der ihn an eine Aktivität in der Zukunft erinnern soll, die erledigt werden soll.
Beteiligt Akteure:	Patient
Auslöser / Vorbedingung:	Der Benutzer befindet sich in der Applikation *despresso* auf der Seite „Kalender“ und klickte auf „neuer Termin“.
Ergebnisse / Nachbedingung:	Ein neuer Termin wird erstellt und im Kalender angezeigt, inklusive einer definierten Erinnerung, die zu gegebener Zeit dem Benutzer angezeigt wird.

Ablauf:

Nr.	Wer	Was
1	Patient	Öffnet Applikation *despresso* und wechselt auf „Kalender“-Seite
2	System	Zeigt Kalender mit vorhandenen Terminen an
3	Patient	Klickt auf „neuer Termin“
4	System	Zeigt leeres Termin-Formular an
5	Patient	Füllt Formular mit Termin-Informationen aus und bestätigt mit „Speichern“
6	System	Speichert den Termin und leitet Benutzer zurück auf die Kalenderseite

Ausnahmen, Varianten:

Nr.	Wer	Was
1	Patient	Benutzer bricht mit „Abbrechen“ ab
2	System	Relevante Informationen fehlen -> Termin kann nicht gespeichert werden

4.2.2 Use Case 2: Stimmung erfassen

Nr. und Name:	2 Stimmung erfassen, evtl. Aktivitätsvorschlag
Szenario:	Patient will / muss Stimmung zwecks Objektivität erfassen.
Kurzbeschreibung:	Patient erfasst momentane Stimmung in immer detaillierteren Graden (gut vs. schlecht -> 40% hoffnungsvoll)
Beteiligt Akteure:	(depressiver) Patient
Auslöser / Vorbedingung:	Homescreen der Applikation offen.
Ergebnisse / Nachbedingung:	Stimmung wird im Kalender erfasst. Evtl. wird eine Aktivität für die erfasste Stimmung vorgeschlagen. In diesem Falle wird Zu-/Absage zur Aktivität auch gespeichert.

Ablauf:

Nr.	Wer	Was
1	Patient	Stimmung gut oder schlecht eingeben.
2	System	Option 1: Eingabe speichern
3	System	Option 2: Eingabe widerrufen
4	System	Option 3: Eingabe verfeinern
5	Patient	Wählt: Eingabe verfeinern
6	System	System schlägt Verfeinerungen vor: z.B. positiv wird unterteilt in hoffnungsvoll, lustig, glücklich, rosig + Neuer Eintrag
7	Patient	Wählt z.B. Eintrag hoffnungsvoll
8	System	Wiederum 3 Optionen: speichern, widerrufen, verfeinern (siehe oben)
9	Patient	Wählt: Eingabe verfeinern
10	System	System schlägt vor, die gewählte Stimmung (hoffnungsvoll) auf einer Skala anzugeben
11	Patient	Wählt auf der „hoffnungsvoll“-Skala 40%
12	System	Option: speichern, widerrufen, Detailtext erfassen
13	Patient	Wählt „Detailtext erfassen“
14	System	Detailtext: Möglichkeit Stimmung als Sprachnachricht, Bildzeichnung, Text oder Bild festzuhalten.
15	Patient	Erfasst Stimmung als fotografiertes Bild
16	Patient	Eingabe „Speichern“
17	System	Speichert Bild unter dieser Stimmung.

Ausnahmen, Varianten:

Nr.	Wer	Was
1.1	Patient	Wählt z.B. „sehr schlecht“.
5.1	Patient	Wählt „speichern“
5.2	System	Zusätzlich zum Speichern schlägt das System vor, der Patient solle sich z.B. einen guten Rooibos Chaï machen. Auswahlmöglichkeiten: Ja / Nein / später / Widerrufen
5.3	Patient	Quittiert Vorschlag mit „Ja“ (und macht sich einen Rooibos Chaï)
5.4	System	Speichert den Vorschlag zusätzlich zu der Stimmung, inkl. Zu-/Absage.

4.2.3 Use Case 3: Vorschläge für Aktivitäten

Nr. und Name:	3: Activity Suggestions
Szenario:	Patient will einen Aktivitätsvorschlag.
Kurzbeschreibung:	Die Applikation bietet dem Patienten die Möglichkeit einen Aktivitätsvorschlag zu bekommen.
Beteiligt Akteure:	Patient
Auslöser / Vorbedingung:	Der Patient hat freie Zeit und möchte eine Beschäftigung.
Ergebnisse / Nachbedingung:	Die Applikation hat dem Patienten einen Aktivitätsvorschlag gegeben.

Ablauf:

Nr.	Wer	Was
1.1	Patient	Öffnet den Reiter Aktivitätsvorschlag.
1.2	System	Gibt dem Patienten eine Liste mit Aktivitätsvorschlägen.
1.3	System	Filtermöglichkeit für Patienten.
1.4	Patient	Filtert die Aktivitäten seinen Bedürfnissen entsprechend.
1.5	Patient	Einen Aktivitätsvorschlag akzeptieren.
1.6	System	Speichert Aktivitätsvorschlag in Kalender.
1.7	Patient	Kann Aktivitätsvorschläge editieren.

Ausnahmen, Varianten:

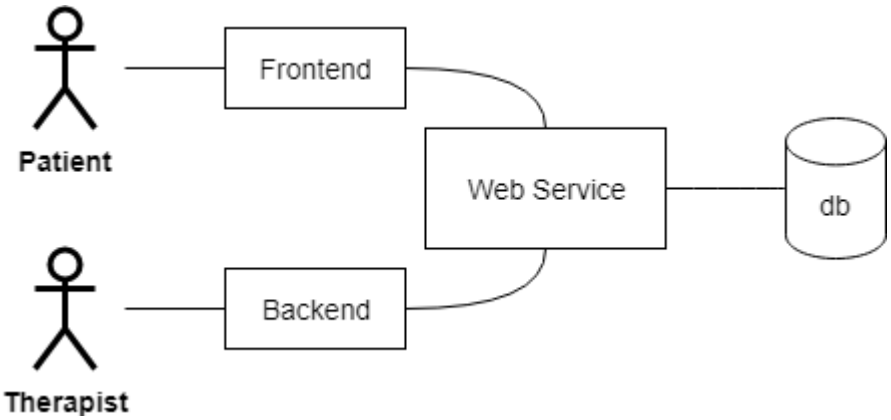
Nr.	Wer	Was
1.5.1	Patient	Lehnt Aktivitätsvorschlag ab. Zurück auf Startbildschirm.
1.6.1	Patient	Speichert den Aktivitätsvorschlag zu einer bestimmten Zeit.
1.6.2	Patient	Führt den Vorschlag sofort aus, ohne in den Kalender zu speichern.

5 System Architecture

Dieses Kapitel gibt eine grobe Übersicht über die Systemkomponenten, die für die Umsetzung der Applikation *despresso* notwendig sind.

Die Anwendung soll Daten auf einer Datenbank speichern, damit von einem Front- und Backend her über die beiden Rollen (Patient und Therapeut) entsprechende Daten abgerufen werden können. Der Therapeut sieht nur Daten von seinen Patienten, die Patientendaten werden untereinander nicht ausgetauscht.

Das Ganze soll für Mobilgeräte optimiert sein, damit der Benutzer einen möglichst kleinen Aufwand hat. Ein Berechtigungssystem soll die Anwendung vor Missbrauch schützen (Therapeut und Patient haben unterschiedliche Berechtigungen).



6 System Requirements Specification

Die Systemanforderungen der Applikation *despresso* lassen sich wie folgt gliedern:

1. Durch die Anforderung, dass sowohl der Patient selber (Mobile und Desktop), wie auch das Pflegepersonal auf gemeinsame Daten zugreifen kann, bietet sich ein Client-Server-Aufbau an.
2. Die Daten werden idealerweise in einer Datenbank auf dem Server abgespeichert.

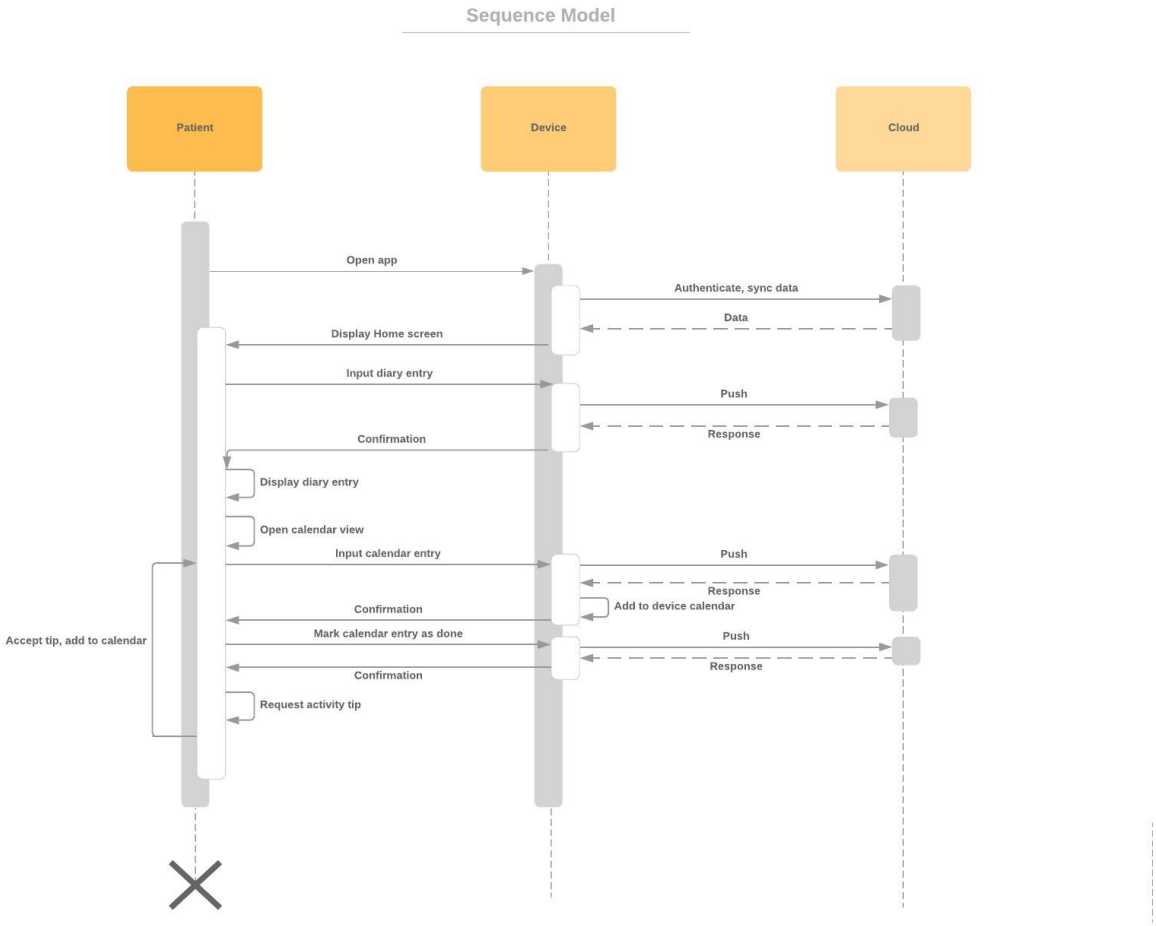
3. Da es sich um heikle Patientendaten handeln kann, welche übertragen und gespeichert werden, ist besonderer Wert auf Sicherheit zu legen.
4. Bisherige Use Cases benötigen keine Schnittstelle zu Umsystemen, jedoch könnte dies in Zukunft ausgeweitet werden.
5. Das System muss skalierbar bei Wachstum sein.

Folgende Detailanforderungen lassen sich zu den oben genannten Grundanforderungen spezifizieren:

1. Der Patient kann Daten eingeben.
    a. Die Daten des Patienten werden auf dem Gerät des Patienten zwischengespeichert.
    b. Sobald eine Verbindung zum Server besteht, werden die Daten hochgeladen.
    c. Aktivitätsbeispiele werden auf dem zentralen Server gespeichert.
    d. Die Aktivitäten werden auch auf dem Endgerät des Patienten gespeichert.
    e. Sobald neue Aktivitäten hinzukommen, werden diese bei Verbindung zum Server mit dem Endgerät synchronisiert.
2. Die Datenbank ist zentral auf einem Server gespeichert.
    a. Die Informationen müssen pro Benutzer gespeichert werden.
    b. Eine Rollenverteilung muss gemacht werden (Patienten sehen nur ihre Daten, Pflegepersonal sieht die Daten ihrer Patienten, Administratoren sehen alle Daten)
3. Der Datenbankserver darf vom Internet nicht erreichbar sein (Firewallzone)
    a. Spezielle Server, welche Internet-facing sind, haben die Berechtigung, auf den Datenbankserver zuzugreifen (Firewall-Regel oder gar Verkabelung)
    b. Der Datentransfer von Client zu Server muss verschlüsselt stattfinden.
    c. Der User muss jederzeit in der Lage sein, all seine Daten zu löschen.
    d. Ein Starkes Passwort muss beim Einrichten erstellt werden.
4. Mögliche Schnittstelen für die Zukunft sind aufs Internet ausgerichtet (z.B.: Wetter, Medikamente, Krankheitsbilder etc.)
5. Die zentrale Datenbank muss leistungsfähig genug sein, um eine zuvor definierte Anzahl an Transaktionen pro Sekunde zu gewährleisten.
    a. Ab einer bestimmten Grösse, muss auch der Datenbankserver duplizierbar sein.
    b. Datenbankserver müssen in der Lage sein, miteinander zu kommunizieren.
    c. Die Internet-facing-Server sind einfach duplizierbar und werden bei mehr Kunden automatisch hochgefahren und angesteuert.

7 System Models

Das Kapitel System Models umfasst die Beschreibung der Beziehung zwischen den Systemkomponenten sowie zwischen dem Gesamtsystem und dessen Kontext. Hier wird am Beispiel eines Kalendereintrags gezeigt, wie die Interaktion zwischen Patient, Device und Cloud aussieht.



8 System Evolution

Dieses Kapitel beschreibt die grundlegenden Annahmen, auf denen das System basiert, und alle erwarteten Änderungen aufgrund von Hardwareevolutionen, sich ändernden Benutzeranforderungen usw. Dieser Abschnitt ist für Systementwickler nützlich, da er ihnen helfen kann, Designentscheidungen zu vermeiden, die mögliche zukünftige Änderungen am System einschränken würden.

Annahmen:

- Programmiersprache: Java.
- Fürs grafische Interface: Vaadin.
- Client-Server Applikation: Daten werden auf einem zentralen Server gespeichert.
- Datenschutz: Eine sichere Verbindung zwischen dem Client und dem Server muss gewährleistet sein.
- Die Applikation muss mobiletauglich sein: Webapp oder Native App.

Mögliche Änderungen:

- Datenschutz kann in Zukunft erleichtert oder erschwert werden.
- Applikation ist zurzeit nur für ein einziges Gerät konzipiert. Könnte in Zukunft auf mehrere Geräte ausgeweitet werden (Fitnessband, Armbanduhr, VR Brille).
- Neben Stimmung, Tipps und Kalendereinträgen könnten in Zukunft weitere Themen hinzukommen.

- Mobile native App entwickeln. Dies würde es ermöglichen offline zu arbeiten.

9 Testing

- Stufe 1: Unit Tests
  Mit Hilfe von Unit Tests soll die Funktionalität der Softwarekomponente sichergestellt werden. Für öffentliche Methoden mit Rückgabewert oder solche die eine öffentlich zugängliche Variable verändern, wird ein Unit Test geschrieben. Unit Tests werden durch die Entwickler geschrieben und während des Entwicklungsprozesses durchgeführt.

- Stufe 2: Systemtests
  Systemtests sollen das ganze System gegen die gesamten Anforderungen (funktionale und nicht funktionale Anforderungen) testen. Systemtests werden mit Testdaten, angelehnt an Originaldaten, durchgeführt. Eine testende Person hält in einem Protokoll fest, welche Resultate die Systemtests geliefert haben.

9.1 Komponenten Tests (Unit-Tests)

Was	Wie
Testziele	Unit Tests überprüfen die erwartete Funktionalität einer Methode. Zu diesem Zweck wird ein erwarteter Wert mit dem tatsächlich aufgetretenen Wert verglichen. Alternativ wird das erwartete Verhalten der Methoden getestet.
Verantwortlichkeit	Entwickler
Testmethode	White Box Test: Der Tester hat Kenntnisse über die Funktionsweise des Systems.
Testabdeckung	Für öffentliche Methoden mit Rückgabewert oder solche die eine öffentliche zugängliche Variable verändern muss ein Unit Test geschrieben werden. Sollte aus technischen Gründen ein Test nicht möglich sein, wird von einem Unit Test abgesehen.
Testumgebung	Der lokale Rechner des Entwicklers dient als Testumgebung.
Tools	IDE (IntelliJ IDEA / Eclipse)
Durchführung	Für jede, zu testende Komponente wird ein Test erstellt. Ist ein Test nicht erfolgreich, wird die Komponente überarbeitet, bis der Test erfolgreich ist.
Auswertung	Die Unit Tests werden nicht speziell protokolliert. Die Überprüfung erfolgt auf dem lokalen Rechner durch den Entwickler.

9.2 Systemtests

Was	Wie
Testziele	Mit den Systemtests wird die Applikation als Ganzes getestet. Es wird überprüft, ob die Applikation wie gewünscht funktioniert und den Anforderungen genügt.
Verantwortlichkeit	Testverantwortlicher
Testmethode	White Box Test: Der Tester hat Kenntnisse über die Funktionsweise des Systems.
Testabdeckung	Die technischen Komponenten dieses Projektes werden getestet.
Testumgebung	Der lokale Rechner des Testenden dient als Testumgebung.
Tools	IDE (falls nötig), Mozilla Firefox (oder ein anderer Browser)
Durchführung	Sobald die erarbeiteten Änderungen implementiert sind, werden Systemtests durchgeführt. Die auszuführenden Testfälle werden zum Voraus definiert.

Auswertung	Für die Systemtests wird ein Testprotokoll erstellt. Dieses wird zusammen mit weiterführenden Informationen der Testperson zur Verfügung gestellt.

9.3 Abnahmetest

Ein Abnahmetest ist für dieses Projekt nicht vorgesehen, da dieser den Umfang übersteigen würde. Es ist geplant, nach der Umsetzung der Applikation ein abschliessendes Feedback bei einer unabhängigen, fachkundigen Person (Interviewpartner) einzuholen und daraus Verbesserungsvorschläge und Weiterentwicklungsmöglichkeiten abzuleiten.

10 Index