Projekt: FitMeUp!

System - Testdokumentation

[Dokumentstruktur basiert auf RUP „Dokument Test Evaluation Summary“]

1 Dokumentinformationen

1.1 Änderungsgeschichte

Datum	Version	Änderung	Autor
08.05.2022	1.0	Erstellung Testplan	Krüger Angelika

1.2 Inhalt

2 Einführung (Introduction)

2.1 Definitionen und Abkürzungen (Definitions, Acronyms, Abbreviations)

GUI = Graphical User Interface

2.2 Referenzen (References)

Literatur: Craig Larman, UML 2 und Patterns angewendet
Vorlesungsunterlagen: Software Engineering (Prof. Dr. A. Doering)

2.3 Übersicht (Overview)

Dieses Dokument soll eine Übersicht über den Aufbau und das Vorgehen der zu testenden Programmteile geben. Die Tests sollen die Einhaltung der Anforderungen überprüfen.

3 Testvorgehen

3.1 Funktionale Tests

3.1.1 Grundtests (Smoke Tests)

Als ersten Schritt ist das Finden von offensichtlichen Fehlern durchzuführen, das mittels explorativen Tests erfolgen soll. So können oberflächliche Tests zu einer schnellen Aussage führen. Mittels explorativer Tests können ca. 35 – 65 % der Fehler gefunden werden. Bevor die Software auf Funktionalität weiter getestet werden soll, werden zuerst Smoke Tests durchgeführt. Diese oberflächliche Überprüfung der Programmfunktionen werden für alle Klassen angewendet. Der erste Smoke Test, der von jedem Entwickler jedes Mal automatisch angewendet wird, ist das Kompilieren der Software. Hier können Änderungen im Quellcode überprüft werden, bevor diese in das Repository eingeht. Wenn hier bereits ein Fehler auftritt, ist es nicht nötig die Software weiter zu testen. Jeder Entwickler testet daher manuell nach einer Kompilierung, ob die Software auf den ersten Blick funktioniert. Bestimmte Merkmale, wie Funktionalität, Interaktion, Präsentation, usw. sollen grob getestet werden. Wenn alle Funktionen einwandfrei durchführbar sind, kann genauer getestet werden, mittels automatischen Tests.

3.1.2 Modul- und Unittests

Alle Modul- und Unittests werden mit dem Unit-Test-Framework GoogleTest durchgeführt. Mit GoogleTest werden alle Klassen, welche die Funktionalitäten der Anwendung abbilden, getestet. Alle Methoden der Klasse werden als Test registriert. Innerhalb der Methoden können weitere Methoden anderer Klassen getestet werden. Dabei werden Tests durchgeführt, die eine Eingabe auf ihre Richtigkeit testen und eine Ausgabe produzieren. Die Dokumentation der Testergebnisse erfolgt über ein Testprotokoll, bei dem jeweils der Testfall und das zugehörige Testergebnis dokumentiert wird. Die Tests sind White-box Tests und orientieren sich daher am Kontrollfluss. Dadurch erreicht man eine gute Testüberdeckung. Bei der Testüberdeckung soll eine 100-prozentige Anweisungsüberdeckung erfolgen. Es wird ein prozentualer Anteil der Anweisung in Test ausgeführt, d.h. jede Anweisung wird mindestens 1x ausgeführt. Des Weiteren soll eine gute Bedingungsüberdeckung erreicht werden. Zusammengesetzte Bedingungen werden in verschiedenen Kombinationen getestet.

3.1.3 Integrationstests

Integrationstests sind auf die Applikation nicht anwendbar. Bei Integrationstests werden die Software-Komponenten mit allen beteiligten Subsystemen auf Funktionalität geprüft. Die Applikation hat keine Verbindung zu anderen Subsystemen.

3.1.4 System Acceptance Test

Alle Tests auf System-/Nutzerebene werden von den einzelnen Teammitgliedern durchgeführt. Jedes Teammitglied testet den Use Case den er umgesetzt hat. Die Tests werden in einem System-Akzeptanz-Dokument festgehalten. Die Ergebnisse der durchgeführten Tests werden dort ebenfalls dokumentiert.

3.2 Bedienbarkeit und Nutzerinterface (Usability)

Die Benutzerfreundlichkeit der FitMeUp!-Applikation soll mithilfe von realen Nutzern getestet werden. Hierbei sollen Interaktionen von Nutzern von den Teammitgliedern beobachtet und dokumentiert werden. Ein Evaluationsbogen, der an die Nutzer verteilt werden soll, soll Ausschluss darüber geben, wie benutzerfreundlich die Applikation ihres Empfindens ist. So kann eine subjektive Einschätzung der Nutzer erhoben werden. Die Nutzergruppe besteht aus zwei Gruppen. In der ersten Gruppe befinden sich Nutzer, die technische Vorkenntnisse besitzen. In der zweiten Gruppe sollen sich Nutzer befinden die kaum technisches Vorwissen haben. Alle ausgewerteten Ergebnisse durch die Beobachtungen der einzelnen Projektteammitgliedern sowie die Evaluationsbogen der Nutzergruppen werden in einem Testprotokoll im Git-Repository hochgeladen.

3.3 Datenschutz, Datensicherheit (Security)

Die Verschlüsselung des Nutzer-Passwort muss korrekt funktionieren. Hier soll die Hashfunktion geprüft werden, da das Passwort aus Sicherheitsgründen nicht im Klartext in die Datenbank gespeichert werden darf.

3.4 Leistungsanforderungen (Performance)

Die Applikation soll auf Leistung geprüft werden. Es soll mittels Tests die Antwortzeit der Applikation durchgeführt werden.

3.5 Zuverlässigkeit

Die Zuverlässigkeit der Applikation muss geprüft werden. Hier ist es wichtig Fehler zu suchen die einen Absturz der Applikation verhindern.

3.6 Schnittstellen

Schnittstellentests sind auf die Applikation nicht anwendbar. Schnittstellentests prüfen das Zusammenspiel der Sub-Systeme bzw. der Komponenten bis hin zum kompletten System. Die FitMeUp!-Applikation stellt ein einziges System dar und hat keine Verbindung zu Subsystemen bzw. weiteren Komponenten.

3.7 Wartung und Servicefunktionen

Eine Log-Datei dient als Wartung und Servicefunktion der Applikation. Dieses wird zur Applikation parallel laufen, um mögliche Fehler und Abstürze zu loggen und rückwirkend verfolgbar zu machen. Die Daten des Loggers werden in einer .txt Datei protokolliert. Hierfür wird das Framework log4cpp für die Programmiersprache C++ genutzt. Diese wird als externe Bibliothek in das Projekt eingebunden.

3.8 Installation

Die Installationstests beziehen sich auf die bereitgestellte Installation. Dieses wird auf Gitlab zur Verfügung gestellt. Hier soll das gesamte Vorgehen der Installation der Applikation getestet werden.

3.9 Internationalisierung / Lokalisierung

Die Internationalisierung und Lokalisierung sind auf die Applikation nicht anwendbar. Die Applikation ist auf den deutschsprachigen Raum beschränkt (siehe Anforderungen).

3.10 Testautomatisierung

Testautomatisierung ist ein automatisiertes Testverfahren und bezeichnet die Automatisierung von Testaktivitäten. Die Testaktivität der Testdurchführung wurde im Beispielprojekt automatisiert. Alle Unit-Tests auf Codeebene, welche die Anforderungen prüfen, werden mithilfe von GoogleTest durchlaufen. Diese liefern ein Ergebnis, welches vom jeweiligen Teammitglied dokumentiert wird.

3.11 Verfolgbarkeit (Traceability)

Hierfür wird ein Excel-Template als Tracebility-Dokument umgesetzt, um alle Anforderungen zu testen und zu verifizieren. Das Dokument ermöglich Identifizierungen der fehlenden Funktionalitäten vorzunehmen. Zusätzlich können Änderungen der Anforderungen aktualisiert werden, sollte das notwendig sein. Mit dem Vorgehen ist eine einfache Verfolgung des Gesamtstatus der Testausführung gewährleistet.

4 Übersicht der Testpläne

Im Laufe der Testverfahren wird ein Test-Dokument erstellt. Darin werden konkrete Testziele spezifiziert. Im Dokument wird eine ID, Use-Case-Nummer, Komponente, Methode und Testverfahren mit Passed/Failed festgehalten. Die konkreten Testschritte und das zu erwartende Ergebnis wird in einem Testcase-Dokument (Excel) dokumentiert. Darin wird eine ID, Prozess, Testcase, Beschreibung und der Status (Passed/Failed) mit dem erwartenden Ergebnis festgehalten.

5 Freigabe von Testergebnissen

Jedes Teammitglied entscheidet zuerst über die Akzeptanz eines Testergebnisses. Bei einer wöchentlich stattfindenden Besprechung werden durchgeführte Tests und deren Testergebnisse nochmals besprochen.