FHNW SQAP-03

Quality Assurance Plan

Version <1.0>

FHNW SQAP-03

Revision History

Date	Version	Description	Author
14.09.2015	0.1	Initial Version	Tobias, Marius, Livio
21.09.2015	0.2	Major Changes to Quality Assurance Plan	Tobias, Livio
21.09.2015	0.3	Reviewed by Marius	Marius
25.09.2015	1.0	Final Draft	Marius

FHNW SQAP-03

Quality Assurance Plan

1. Introduction and Purpose

Zweck des Testplans ist es, ein organisiertes und systematisches Testen des Systems Movieplex7 zu ermöglichen. Es werden die zu testenden Komponenten des Systems benannt und zu diesen adäquate Testfälle entworfen. Auch werden Kriterien gesetzt, wann der Test als erfolgreich durchgeführt gilt und wann er abgebrochen werden muss.

Desweiteren werden Anforderungen an Hardware, Software und andere Betriebsmittel, die für die Tests benötigt werden, definiert.

Der Testplan richtet sich an den Kunden, an das Entwicklungsteam, insbesondere die Tester.

1.1 Scope

Der Umfang dieses Projekts beschränkt sich auf das Projekt Movieplex7:

- Das Projekt hat keine direkte Abhängigkeiten zu anderen Projekten.
- Die Applikation wird auf einer JVM 1.8 ausgeführt 64- oder 32-bit

Schwerpunkt dieses Testplans sind die Integration-, Stress- sowie Leistungsstests. Komponententests sind in Form von Unittests vorhanden (JUnit) und werden hier nicht weiter dokumentiert. Es werden keine Usability tests durchgeführt

1.2 Definitions

Stresstest

Test, welcher die Robustheit einer Applikation testet gegen normale Operationen und Verfügbarkeit . Der Test stellt die Standard-Operationen einer Applikation sicher.

Testabdeckung (durch Unittests)

Als Testabdeckung bezeichnet man das Verhältnis an tatsächlich getroffenen Aussagen eines Tests gegenüber den theoretisch möglich treffbaren Aussagen bzw. der Menge der gewünschten treffbaren Aussagen. Die Testabdeckung spielt als Metrik zur Qualitätssicherung und zur Steigerung der Qualität insbesondere im Maschinenbau und der Softwaretechnik eine große Rolle.

Junit
JUnit ist ein Framework zum Testen von Java-Programmen, das besonders für automatisierte Unit-Tests einzelner Units (Klassen oder Methoden) geeignet ist.

Mockito
Mockito ist eine Programmbibliothek zum Erstellen von Mock-Objekten für Unit-Tests von Java-Programmen.

Regression Tests
Eine Test-Methode mit der versucht wird, neue Bugs einer Applikation zu finden. Es wird unterschieden zwischen “Automated Regression Testing” und “Manual Regression Testing” **.**

Code-Coverage
Wie viel Code wird von der Applikation tatsächlich durchlaufen

Test-Coverage
Welche Teile der Applikation werden von Testfällen abgedeckt

Database Integrity Testing

Es werden die einzelnen Subsysteme einer Datenbank getestet. Datenbanken werden als unabhängiges Subsystem getestet.

Unit testing

Alle Use-Cases werden in kleinsten Einheiten (Units) getestet. Absicht ist die Verifizierung von Daten-Akzeptanz in der Business-Logik. Die Cases werden gegen eine Blackbox getestet.

Performance testing

Test, bei dem Reaktions-Zeiten, Übertragungs-Zeiten und die Funktionalität von Hardware-Komponenten gemessen werden. Die zu erreichenden Ziele sind als funktionale Anforderungen im Pflichtenheft definiert.

Git

Git ist eine freie Software zur verteilten Versionsverwaltung von Dateien**.**

1.3 Acronyms, and Abbreviations

JVM	*Java Virtual Machine*
RE	*Requirements Engineer*
TM	*Test Manager*
UM	*Usability Manager*
QAP	*Quality Assurance Plan*

1.4 References

- Google Coding Convention: https://google.github.io/styleguide/javaguide.html

FHNW SQAP-03

1. Management

1.5 Tasks and Responsibilities

Task	*Responsibility*
Prozess-Verifizierung	*RE*
Systemgrenzen, Systemkontext	*RE*
Erstellen STQM	*TM*
Erstellen Testplan/Testfälle	*TM*
Auswertung Kundenbewertung	*RE*

2. Standards and Guidelines

2.1 Checkstyle

Der im Zuge dieses Projekts produziert Quellcode muss konform mit den Standard Checkstyle Regeln sein.

2.2 Coding Conventions

Der im Zuge dieses Projekts produziert Quellcode muss konform mit dem Google Styleguide sein: *Siehe References.*

3. Metrics

- Testabdeckung in Prozent
- Code-Coverage in Prozent
- Anzahl Bugs (visuell)
- Reaktions- und Ladezeiten der Applikation
- Build-Fails (Jenkins)

4. Review

4.1 Peer Reviews

Es wird das Jenkins-Build-System verwendet. Jeder Benutzer erhält eine Email-Nachricht, sobald ein Build fehlschlägt. Für die Qualität eines Builds ist der Entwickler zuständig. Der Qualitätssicherungs-Manager gibt Richtlinien vor.

4.2 Pair Programming

Zunächst soll Paarprogrammierung die Softwarequalität steigern. Durch die Kontrollfunktion der zweiten Person sollen problematische Lösungen vermieden werden. Die Paarprogrammierung dient aber auch zur Verbreitung von Wissen über den Quellcode. Durch das regelmäßige Rotieren der Partner kann immer der

jeweils neue Partner durch Learning by Doing etwas über die bearbeiteten Quelltexte lernen.

5. Tools, Techniques, and Methodologies

- Junit
- Mockito
- Manual Regression Tests
- Code-Coverage
- Database Integrity Testing
- Test Script Automation Tools
- Git (Versionsverwaltung)
- Jenkins (Monitoring- und Buildsystem)
- Datenbank-Utilities
- Data-Generator Tool

6. Testing

Database

- *Abfragen der Datenbank und Datenbank-Subsysteme mit gültigen und ungültigen Daten*
- *Testen der Datenbank-Relationen in der dritten Normalform*
- *Testen von Datenbank-Credentials*

Unit-Tests

- *Die Unittests sind zu finden unter **movieplex7/test/java***
- *Testen von Units anhand Testdaten*
- *Die Qualität der Unit-Tests wird anhand der Test-Coverage gemessen*

Integration-Tests

- *Testen von Applikationsblöcken und definierten Abläufen*
- *Zusammenspiel einzelner Units in der Business-Logik*
- *Performance-Tests*

GUI-Tests

- *Werden an dieser Stelle nicht durchgeführt.*

Performance-Tests

- Reaktionszeiten durch Erhöhen der Transaktionen und durch Erstellen von virtuellen Benutzern (Datenbank und Applikation)
- Übertragungszeiten anhand Netzwerk-Monitoring-Tools

- Testen von Hardware-Komponenten bezüglich Geschwindigkeit anhand Hardware-Monitoring-Tools

Load Tests

- Messen von Peaks in unterschiedlichen Environments

Stress Tests

- Innerhalb Systemgrenze
- Erhöhen der Transaktionen

7. Deliverables

7.1 Quality Reports

- *Kontinuierlich via Jenkins (Checkstyle Regelverstösse, Failing JUnittests)*

7.2 Work Products

- *Testplan: Definiert die konkreten Testfälle. → Verantwortung TM.*

7.3 Automation

Automatisierung soll wo möglich angestrebt werden:

- Maven (pom.xml)
- Jenkins

7.4 Test Guidelines

- *Sowohl Blackbox als auch Whitebox Tests sollen durchgeführt werden.*
- *Tests sollten wo möglich automatisiert werden (Scripts, Jenkins etc.)*
- *Tests sollten regelmässig / iterativ durchgeführt werden.*