Projekt: VirtualCalendar

System - Testdokumentation

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

1 Dokumentinformationen

1.1 Änderungsgeschichte

Datum	Version	Änderung	Autor
11.05.2017	1	Erstellung	Artinger
31.05.2017	2	Bearbeitung nach Review	Artinger

1.2 Inhalt

2 Einführung (Introduction)

2.1 Übersicht (Overview)

In diesem Dokument wird unser Testvorgehen beschrieben.

3 Testvorgehen

3.1 Funktionale Tests

3.1.1 Grundtests (Smoke Tests)

Beim Kompilieren im Debug-Modus werden offensichtliche Fehler erkannt und können schnell behoben werden. Deshalb wird vor dem Hochlanden ins Repository das Programm im Debug-Modus ausgeführt.

3.1.2 Modul- und Unittests

Getestet werden unsere einzelnen Module in Visual Studio mit NUnit. Mit diesem Programm wird eine automatische Dokumentation erzeugt. Dafür verantwortlich sind Herr Sanchez und Frau Draghiciu.
Außerdem werden wir manche Module auch manuell testen, dabei lassen wir unser Programm mehrmals durchlaufen und geben unterschiedliche Eingaben ein. Es muss auch auf falsche Eingaben getestet werden. Dokumentiert werden Anzahl der Durchläufe, Erwartetes Ergebnis, richtige oder falsche Eingabe und das endgültige Ergebnis.
Am Ende muss entschieden werden, ob das Ergebnis zufriedenstellend ist oder ob noch Änderungen vorzunehmen sind.

3.1.3 Integrationstests

Da bei unserem Programm mehrere Seiten zusammengefügt werden müssen, sind Integrationstest notwendig.
Wenn verschiedene Module zusammengefügt werden, muss nochmal getestet werden, ob diese Module korrekt verbunden sind und auch zusammenarbeiten.

3.1.4 System Acceptance Test

Hier eine Übersicht über unsere Tests, für eine ausführliche Dokumentation siehe:
https://fbim.hs-regensburg.de/svn/doering_swp_ss2017_team02/Produkt/Test

Inhalt	Durchführung	Verantwortlicher
Profil anlegen/bearbeiten	Manuell	Kuczera
Termin erstellen	Manuell	Kuczera
Termin löschen		
Erinnerung einstellen		
Gruppe erstellen und Einladungen verschicken	Manuell	Kuczera
Datenbank		Sanchez
Anmelden	Manuell	Kuczera

Kompatibilität der einzelnen Ansichten	Manuell	Draghiciu
GUI		

3.2 Bedienbarkeit und Nutzerinterface (Usability)

Dies wird getestet, sobald alle Seiten miteinander verbunden sind, um zu sehen, ob unser Programm einfach zu verstehen und auch logisch aufgebaut ist.

3.3 Datenschutz, Datensicherheit (Security)

Dies testen wir auch anhand von automatischen Tests mit NUnit.

3.4 Leistungsanforderungen (Performance)

Bei automatisierten Test wird die Zeit aufgenommen, die das Programm braucht, um Funktionen auszuführen, dabei muss geschaut werden, dass dies nicht zu lange dauert.

3.5 Zuverlässigkeit

Alle Tests werden mehrfach durchgeführt, damit man sieht, ob immer das erwartete Ergebnis erzielt wird und nicht immer unterschiedliche Ergebnisse zustande kommen.

3.6 Schnittstellen

Es müssen die Schnittstellen zur Datenbank und zum E-Mail-Benachrichtigungssystem getestet werden. Dabei wird bei der Datenbank getestet, ob man Daten hineinschreiben, bearbeiten und auch wieder auslesen kann. Das kann man an der Datenbank mit verfolgen, wenn man sich die Tabellen ansieht.
Die Schnittstelle zum E-Mail-Benachrichtigungssystem wird getestet, in dem man sich E-Mails senden lässt und dann überprüft, ob diese ankommen und der richtigen Inhalt gesendet wurde.

3.7 Installation

Die Installation wird getestet, sobald eine Betaversion vorliegt. Dabei wird eine ausführbare Datei erzeugt und diese dann auf einem Rechner installieren, auf welchem das Programm noch nicht vorhanden ist.

3.8 Testautomatisierung

Wir verwenden NUnit für automatisierte Tests.

3.9 Verfolgbarkeit (Traceability)

Wir testen erst kleiner Abschnitte und wenn diese dann fehlerfrei funktionieren, dann werden die Teilstücke zusammengesetzt. Somit weiß man dann, dass nach dem Zusammenführen nur Fehler in der Kompatibilität liegen können.

4 Übersicht der Testpläne

- Pfad: https://fbim.hs-regensburg.de/svn/doering_swp_ss2017_team02/Produkt/Test

5 Freigabe von Testergebnissen

Mindestens zwei Mitglieder testen zusammen, damit man sich beraten kann, ob dies schwerwiegende Fehler sind oder eher leicht zu behebende Fehler.
Wenn beide dann zur gleichen Entscheidung kommen, dass der Fehler minimal ist, wird das Testergebnis akzeptiert.

Bei automatischen Test gibt NUnit an, ob es akzeptiert wird oder nicht.