I8 - Erstellung und Verwaltung von Gruppen für Software Engineering Projekte - Testdokumentation

Inhaltsverzeichnis

1. Allgemeine Informationen

In der Testdokumentation werden die erstellten Testobjekte bezogen auf ihre Test Cases beschrieben und die Funktionalität der einzelnen Softwarekomponenten, als auch das Zusammenspiel dieser überprüft. Ziel ist es möglichst viele Auftretende Ereignisse vor der Übergabe an den Auftraggeber zu überprüfen, um sicherzustellen das die einzelnen Komponenten den erwartetem Verhalten entsprechen. Es sollen vor allem auch Situationen simuliert werden, in denen die Website mit fehlerhaften Eingaben gefüttert wird, um zu überprüfen, ob der Nutzer eine Warnung oder Fehlermeldung erhält, welche ihn auffordert seine Eingabe zu ändern.

2. Testprotokoll Login

Test Cases:

- Test Case 1: Login-Felder werden überprüft
- Test Case 2: Login für Student funktioniert
- Test Case 3: Login für Dozent funktioniert
- Test Case 4: Login mit falschen Daten schlägt fehl

Testobjekt:

- Login.html
- Datenbank Entity Users

Erwartung:

- Login für Studenten und Dozenten
- Login für Studenten ->Weiterleitung auf den Fragebogen der Studentenansicht
- Login für Dozenten -> Weiterleitung auf die Dozenten-Website
- Abfrage der Login Daten aus Datenbank

Eingabedaten:

Positiv Test

```
@Test
    public void testLoginPageWithDatabaseCredentials() {
        // Query, um eine der Benutzername- und Passwortkombinationen aus der Datenbank
abzurufen
        String query = "SELECT username, password FROM users LIMIT 1";

        try {
            Statement statement = connection.createStatement();
```

```
            ResultSet resultSet = statement.executeQuery(query);

            // Falls ein Datensatz vorhanden ist, verwenden Sie den ersten Benutzernamen
und das erste Passwort
            if (resultSet.next()) {
                String expectedUsername = resultSet.getString("username");
                String expectedPassword = resultSet.getString("password");

                // Act
                driver.get("Docker/Fronten/login.html");

                WebElement usernameInput = driver.findElement(By.id("username"));
                WebElement passwordInput = driver.findElement(By.id("password"));
                WebElement loginButton = driver.findElement(By.tagName("button"));

                usernameInput.sendKeys(expectedUsername);
                passwordInput.sendKeys(expectedPassword);
                loginButton.click();

                // Assert
                String pageTitle = driver.getTitle();
                Assert.assertEquals("Erwarteter Seitentitel", pageTitle);


            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
```

Negativ Test

```
@Test
    public void testLoginPageWithDatabaseCredentials() {
        // Query, um eine der Benutzername- und Passwortkombinationen aus der Datenbank
abzurufen
        String query = "SELECT username, password FROM users LIMIT 1";

        try {
            Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery(query);

            // Falls ein Datensatz vorhanden ist, verwenden Sie den ersten Benutzernamen
und das erste Passwort
            if (resultSet.next()) {
                String expectedUsername = resultSet.getString("username");
                String expectedPassword = resultSet.getString("password");

                // Act
                driver.get("Docker/Fronten/login.html");

                WebElement usernameInput = driver.findElement(By.id("username"));
                WebElement passwordInput = driver.findElement(By.id("password"));
                WebElement loginButton = driver.findElement(By.tagName("button"));

                usernameInput.sendKeys(expectedUsername);
                passwordInput.sendKeys(expectedPassword);
```

```
                loginButton.click();

                // Assert
                String pageTitle = driver.getTitle();
                Assert.assertEquals("Erwarteter Seitentitel", pageTitle);

            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
```

Testvorgang:

- Ausfüllen der beiden Felder
- Absenden über den Login-Button

Ergebnis:

- Bei Eingabe der richtigen Daten wird man wie erwartet auf die jeweilige Seite weitergeleitet
- Bei Eingabe von falschen Daten bekommt man eine Fehlermeldung, die den Nutzer auffordert die eingegebenen Daten zu überprüfen

Test war <u>erfolgreich</u>.

3. Testprotokoll Projekt anlegen Test1

Modultest

Test Cases:

- Test Case 1: Überprüfen des Wertes Projetkt_ID
- Test Case 2: Überprüfen des Wertes ProjektName
- Test Case 3: Überprüfen des Wertes Firma
- Test Case 4: Überprüfen des Wertes Description

Testobjekt:

- Projektverwaltung.html
- Datenbank Entity Projekte

Erwartung:

- Abspeichern der angelegten Projekte
- Projekt ID als Primäre Schlüssel zur eindeutigen Beschreibung
- Pflichtfelder Hinweise bei Nichtausfüllen

Eingabedaten:

```
@Test
    public void testProjectFields() {
        // Arrange
        Set<String> projectIDs = new HashSet<>();
        Project project = new Project();
       // Act
        project.setProjectID("I1");
        project.setProjectName("Test Project1");
        project.setFirma("Herr Müller");
        project.setDescription("This is a sample project.");

        // Assert
        Assert.assertTrue(projectIDs.add(project.getProjectID())); // Die erste ID sollte
einzigartig sein

        // Act (erneut)
        project.setProjectID("I1"); // Versuch, dieselbe ID erneut zu setzen
        project.setProjectName("Test Project2");
        project.setFirma("Frau Meier");
        project.setDescription("This is another test project.");

        // Act (erneut)
```

```
        project.setProjectID("I3"); // Versuch, dieselbe ID erneut zu setzen
        project.setProjectName("Test Project3");
        project.setFirma("Frau Mustermann");
        project.setDescription("");

        // Assert (sollte fehlschlagen)
        Assert.assertFalse(projectIDs.add(project.getProjectID())); // Die zweite ID sollte
bereits existieren und kann nicht hinzugefügt werden

        // Weitere Assertions für die anderen Felder
        Assert.assertEquals("Test Project1", project.getProjectName());
        Assert.assertEquals("Herr Müller", project.getFirma());
        Assert.assertEquals("This is a sample project.", project.getDescription());
    }
```

Testvorgang:

- Ausfüllen der Maske
- Betätigen des Speichern Buttons
- Überprüfen der Daten in der Datenbank

Ergebnis:

- Eingabe der Daten funktioniert
- Projekte werden in der Datenbank gespeichert
- Projekte werden auf der Website in einer Tabelle angezeigt
- Es ist möglich 2 mal die gleiche ID für verschiedene Projekte zu verwenden
- Auch bei leeren Felder wird das Projekt erstellt

Test war <u>nicht erfolgreich</u>.

Bei einem erfolgslosen Test, Programmierern Protokoll schicken.

Hinweis zur Optimierung:

- Festlegen, dass die ID nur einmal verwendet werden darf
- Pflichtfelder mit Aufforderung zum Ausfüllen aller Felder

4. Testprotokoll Projekt anlegen Test2

Modultest

Test Cases:

- Test Case 1: Überprüfen des Wertes Projetkt_ID
- Test Case 2: Überprüfen des Wertes ProjektName
- Test Case 3: Überprüfen des Wertes Firma
- Test Case 4: Überprüfen des Wertes Description

Testobjekt:

- Projektverwaltung.html
- Datenbank Entity Projekte

Erwartung:

- Abspeichern der angelegten Projekte
- Projekt ID als Primäre Schlüssel zur eindeutigen Beschreibung
- Pflichtfelder Hinweise bei Nichtausfüllen

Eingabedaten:

```
@Test
    public void testProjectFields() {
        // Arrange
        Set<String> projectIDs = new HashSet<>();
        Project project = new Project();
       // Act
        project.setProjectID("I1");
        project.setProjectName("Test Project1");
        project.setFirma("Herr Müller");
        project.setDescription("This is a sample project.");

        // Assert
        Assert.assertTrue(projectIDs.add(project.getProjectID())); // Die erste ID sollte
einzigartig sein

        // Act (erneut)
        project.setProjectID("I1"); // Versuch, dieselbe ID erneut zu setzen
        project.setProjectName("Test Project2");
        project.setFirma("Frau Meier");
        project.setDescription("This is another test project.");

        // Act (erneut)
        project.setProjectID("I3"); // Versuch, dieselbe ID erneut zu setzen
        project.setProjectName("Test Project3");
```

```
        project.setFirma("Frau Mustermann");
        project.setDescription("");

        // Assert (sollte fehlschlagen)
        Assert.assertFalse(projectIDs.add(project.getProjectID())); // Die zweite ID sollte
bereits existieren und kann nicht hinzugefügt werden

        // Weitere Assertions für die anderen Felder
        Assert.assertEquals("Test Project1", project.getProjectName());
        Assert.assertEquals("Herr Müller", project.getFirma());
        Assert.assertEquals("This is a sample project.", project.getDescription());
    }
```

Testvorgang:

- Ausfüllen der Maske
- Betätigen des Speichern Buttons
- Überprüfen der Daten in der Datenbank

Ergebnis:

- Eingabe der Daten funktioniert
- Projekte werden in der Datenbank gespeichert
- Projekte werden auf der Website in einer Tabelle angezeigt
- Es ist nicht möglich 2-mal die gleiche ID für verschiedene Projekte zu verwenden
- Fehler mit Hinweis zum Ändern der ID
- Fehlermeldung bei nicht ausgefüllten Feldern

Test war <u>erfolgreich</u>.

5. Testprotokoll Fragebogen

Modultest

Test Cases:

- Test Case 1: Überprüfen des Wertes Skill
- Test Case 2: Überprüfen des Wertes ProjectChoice
- Test Case 3: Überprüfen des Wertes ActivityChoice
- Test Case 4: Überprüfen des Wertes Fachinformatiker

Testobjekt:

- Studentenansicht.html
- Datenbank Entity Role
- Datenbank Entity Project
- Datenbank Entity Skill
- Datenbank Entity skillanswer
- Datenbank Entity rolequestion
- Datenbank Entity projectquestion

Erwartung:

- Abspeichern der Daten beim Absenden mittels des „Absenden“ Buttons
- Auch ein unvollständig ausgefüllter Fragebogen kann gespeichert werden
- Beim erneuten Anmelden werden die Daten wieder im Fragebogen angezeigt

```
@Test
    public void testFillOutQuestionnaire() {

        // Act
        driver.get("file://../Docker/Frontend/Student/Studentenansicht.html");

        WebElement projectPreferenceChoice = driver.findElement(By.id("choice2"));
        WebElement activityPreferenceChoice = driver.findElement(By.id("choice3"));
        WebElement skillsChoice = driver.findElement(By.id("choice4"));
        WebElement sendButton = driver.findElement(By.id("sendQuestionnaire"));

        // Fragebogen ausfüllen
        projectPreferenceChoice.click();
        activityPreferenceChoice.click();
        skillsChoice.click();
        sendButton.click();


    }
```

Testvorgang:

- Ausfüllen des Fragebogens
- Absenden des Fragebogens
- Erneutes Anmelden zum Überprüfen der angezeigten Daten

Ergebnis:

- Fragebogen wird richtig gespeichert
- Auch bei erneutem Anmelden werden die bereits ausgefüllten Punkte wieder angezeigt
- Auch bei nicht komplett ausgefüllten Fragebogen wird der bereits beantwortete Teil gespeichert

Test war <u>erfolgreich</u>.

6. Testprotokoll Teams generieren

Integrationstest

Beschreibung:

Der Algorithmus soll eine Erweiterung des bereits bestehenden sein. Der bereits bestehende Algorithmus liefert beim Wiederholen mit gleichen Testdaten kein endgültiges, konsistentes Ergebnis, da es immer das möglichst beste Ergebnis liefert. Die einzige Möglichkeit den Algorithmus zu testen, ist das Prüfen der Restriktionen, die das Ergebnis beschreiben.

Testobjekt:

- Teams_generieren.html
- Algo.java Datei
- verwendete Entitys der run()-Funktion

Erwartung:

- Generierung der Teams basierend auf den Antworten des Fragebogen
- Einbeziehung der Daten aus der ausgelesenen CSV-Datei, gespeichert in der Entity „student“
- Ausgabe von Teams mit der berücksichtigten Mindestanzahl welche der Dozent festgelegt hat
- Alle Studenten wurden berücksichtig
- Optimale Nutzung der Projekte
- Gleichmäßige Verteilung der Studenten, besonders der Wirtschaftsingenieure

Testvorgang:

- Eingabe der Mindestanzahl
- Starten des Algorithmus über den „ Teams generieren“-Button

Ergebnis:

- Keine Ausgabe
- Der Algorithmus liefert keine Teams
- Die Restriktion und auch das A bgreifen der Daten aus der Datenbank konnte nicht überprüft werden

Test war <u>nicht erfolgreich</u>.

Hinweis:

- Aufrufen der Funktion ist fehlgeschlagen
- Kein weiterer Test möglich