**Testdokumentation – Webapplikation „Multiuserapp“**

---

**1. Testkonzept**

Das Ziel des Testkonzepts ist es, die Qualität und Funktionalität der Webapplikation „Multiuserapp“ sicherzustellen.
Es werden verschiedene Testarten eingesetzt, um Fehler frühzeitig zu erkennen und die Zuverlässigkeit des Systems zu gewährleisten.
Die Tests decken alle Ebenen der Anwendung ab: von einzelnen Funktionen bis hin zum Gesamtsystem.

---

**2. Teststrategie**

Die Teststrategie umfasst folgende Ebenen:

- **Unittests:** Überprüfen einzelne Methoden und Funktionen der Backend- und Frontend-Logik.
- **Komponententests:** Testen einzelne Komponenten (z.B. React-Komponenten, Controller) isoliert.
- **Integrationstests:** Testen das Zusammenspiel mehrerer Komponenten oder Systeme (z.B. Backend mit Datenbank).
- **Systemtests:** Testen die gesamte Anwendung aus Sicht des Endnutzers (End-to-End).

Die Tests werden sowohl **manuell** als auch **automatisiert** durchgeführt.

---

**3. Testinfrastruktur**

- **Backend:** Java Spring Boot, getestet mit JUnit und Mockito
- **Frontend:** React, getestet mit Jest und React Testing Library
- **Datenbank:** MySQL, für Integrationstests mit Testdatenbank
- **Testumgebung:** Docker-Container für konsistente Testumgebung
- **CI/CD:** Automatisierte Tests über GitHub Actions (optional)

---

**4. Testarten**

- **Unittest:** Testet einzelne Funktionen oder Methoden isoliert, z.B. Validierung einer E-Mail-Adresse

- **Komponententest:** Testet eine komplette Komponente (z.B. React-Komponente oder Spring Controller) ohne externe Abhängigkeiten
- **Integrationstest:** Testet das Zusammenspiel mehrerer Komponenten, z.B. Controller mit Service und Datenbank
- **Systemtest:** Testet die gesamte Anwendung als Blackbox aus Sicht des Nutzers

---

**5. Testfälle**

---

**Nr. 1**

- **Testfall:** Registrierung eines neuen Users
- **Testart:** Systemtest
- **Erwartetes Ergebnis:** User wird erfolgreich registriert und kann sich einloggen
- **Testergebnis:** Erfolgreich
- **Fehlerursache (falls fehlgeschlagen):** -
- **Behoben?:** -
- **OWASP Schwachstelle / Schnittstelle:** -

---

**Nr. 2**

- **Testfall:** Login mit falschem Passwort
- **Testart:** Systemtest
- **Erwartetes Ergebnis:** Fehlermeldung „Falsches Passwort“ erscheint, kein Login möglich
- **Testergebnis:** Erfolgreich
- **Fehlerursache (falls fehlgeschlagen):** -
- **Behoben?:** -
- **OWASP Schwachstelle / Schnittstelle:** -

---

**Nr. 3**

- **Testfall:** Doppelte Registrierung mit gleicher E-Mail

- **Testart:** Integrationstest
- **Erwartetes Ergebnis:** Fehlermeldung „E-Mail bereits vergeben“ erscheint
- **Testergebnis:** Fehlgeschlagen
- **Fehlerursache (falls fehlgeschlagen):** Backend prüfte E-Mail nicht eindeutig
- **Behoben?:** Ja, Eindeutigkeit im Backend ergänzt
- **OWASP Schwachstelle / Schnittstelle:** -

---

**Nr. 4**

- **Testfall:** Raumübersicht wird geladen
- **Testart:** Komponententest
- **Erwartetes Ergebnis:** Alle verfügbaren Räume werden angezeigt
- **Testergebnis:** Erfolgreich
- **Fehlerursache (falls fehlgeschlagen):** -
- **Behoben?:** -
- **OWASP Schwachstelle / Schnittstelle:** -

---

**Nr. 5**

- **Testfall:** Reservierung eines Raums an einem belegten Termin
- **Testart:** Integrationstest
- **Erwartetes Ergebnis:** Fehlermeldung „Raum bereits reserviert“ erscheint
- **Testergebnis:** Erfolgreich
- **Fehlerursache (falls fehlgeschlagen):** -
- **Behoben?:** -
- **OWASP Schwachstelle / Schnittstelle:** -

---

**Nr. 6**

- **Testfall:** Bearbeiten einer bestehenden Reservierung
- **Testart:** Systemtest
- **Erwartetes Ergebnis:** Reservierung kann geändert und gespeichert werden

- **Testergebnis:** Erfolgreich
- **Fehlerursache (falls fehlgeschlagen):** -
- **Behoben?:** -
- **OWASP Schwachstelle / Schnittstelle:** -

---

**Nr. 7**

- **Testfall:** Löschen einer Reservierung
- **Testart:** Komponententest
- **Erwartetes Ergebnis:** Reservierung wird entfernt und ist nicht mehr sichtbar
- **Testergebnis:** Fehlgeschlagen
- **Fehlerursache (falls fehlgeschlagen):** Frontend aktualisierte Liste nicht automatisch
- **Behoben?:** Ja, automatische Aktualisierung implementiert
- **OWASP Schwachstelle / Schnittstelle:** -

---

**Nr. 8**

- **Testfall:** Zugriff auf „Meine Buchungen“ ohne Login
- **Testart:** Systemtest
- **Erwartetes Ergebnis:** Weiterleitung zur Login-Seite
- **Testergebnis:** Erfolgreich
- **Fehlerursache (falls fehlgeschlagen):** -
- **Behoben?:** -
- **OWASP Schwachstelle / Schnittstelle:** -

---

**Nr. 9**

- **Testfall:** Admin-Login mit falschen Daten
- **Testart:** Systemtest
- **Erwartetes Ergebnis:** Fehlermeldung „Ungültige Zugangsdaten“ erscheint
- **Testergebnis:** Erfolgreich

- **Fehlerursache (falls fehlgeschlagen):** -
- **Behoben?:** -
- **OWASP Schwachstelle / Schnittstelle:** -

---

**Nr. 10**

- **Testfall:** Admin bearbeitet Reservierung eines Users
- **Testart:** Integrationstest
- **Erwartetes Ergebnis:** Änderungen werden gespeichert und sind für User sichtbar
- **Testergebnis:** Fehlgeschlagen
- **Fehlerursache (falls fehlgeschlagen):** Admin-Änderungen wurden nicht an User-Ansicht übermittelt
- **Behoben?:** Ja, Synchronisation nachgebessert
- **OWASP Schwachstelle / Schnittstelle:** -

---

**Nr. 11**

- **Testfall:** Zugriffsschutz laut OWASP (Broken Access Control)
- **Testart:** Sicherheitstest (OWASP)
- **Erwartetes Ergebnis:** Nicht authentifizierte User dürfen keine geschützten Seiten/Funktionen sehen
- **Testergebnis:** Fehlgeschlagen
- **Fehlerursache (falls fehlgeschlagen):** Zugriffskontrolle im Backend fehlte, User konnten Admin-Seiten aufrufen
- **Behoben?:** Ja, Zugriffskontrolle nach OWASP-Empfehlung implementiert
- **OWASP Schwachstelle / Schnittstelle:** A01:2021 – Broken Access Control
  Diese Schwachstelle tritt auf, wenn Benutzer auf Funktionen oder Daten zugreifen können, für die sie keine Berechtigung haben. Laut OWASP Top Ten ist dies eine der kritischsten Sicherheitslücken in Webanwendungen.