Testkonzept

Getestet wird, um sicherzustellen, dass eine Software-Anwendung korrekt funktioniert, Benutzeranforderungen erfüllt und frei von Fehlern ist, die die Nutzung beeinträchtigen könnten, sowie um das Risiko von Problemen nach der Veröffentlichung zu minimieren und dadurch die Qualität und Zuverlässigkeit des Produktes zu gewährleisten.

## Testmethoden

**Blackbox-Test**: Hierbei kennt der Tester den internen Aufbau oder den Code der Anwendung nicht und testet nur die Benutzeroberfläche und die Funktionalitäten, um sicherzustellen, dass die Anforderungen erfüllt sind.

**Whitebox-Test**: Im Gegensatz zum Blackbox-Test hat der Tester Zugang zum internen Code und kann so die internen Strukturen testen, um die Abdeckung des Codes sicherzustellen, beispielsweise durch Pfad- oder Zweigtests.

**Unit-Test**: Dabei werden einzelne Komponenten oder Funktionen der Software isoliert getestet, um ihre korrekte Funktionsweise zu verifizieren.

**Integrationstest**: Hierbei werden mehrere Komponenten oder Systeme kombiniert und als Gruppe getestet, um zu überprüfen, ob sie korrekt zusammenarbeiten.

**Systemtest**: Bei dieser Art des Tests wird das gesamte, vollständig integrierte System auf Übereinstimmung mit den spezifizierten Anforderungen geprüft.

**Akzeptanztest**: Dieser Test wird durchgeführt, um festzustellen, ob das System die Geschäftsanforderungen erfüllt und für den Einsatz beim Endbenutzer bereit ist.

## Verwendete testmethoden

In meiner Angular-Projektentwicklung setze ich Whitebox-Testing und Unit-Tests um, indem ich die mit Angular vorinstallierten Testpakete nutze. Für Whitebox-Testing verwende ich das Jasmine Test Framework, welches tiefe Einblicke in die interne Struktur des Codes ermöglicht. Jasmine erlaubt es mir, Testfälle zu definieren und Erwartungen für Funktionen und Methoden festzulegen. Zusammen mit dem Angular Testbed, ein dynamisches Testmodul, das das Testen von Angular-spezifischem Code wie Komponenten, Services und Pipes erleichtert, ermöglicht es mir, die Funktionalität meiner Anwendung auf Einheiten-Ebene zu überprüfen. Für die Ausführung der Unit-Tests benutze ich Karma, den Test Runner, der die Testfälle in einer realen Browserumgebung ausführt und Feedback in Echtzeit bietet. Diese Werkzeuge stellen sicher, dass jeder Aspekt meiner Anwendung gründlich geprüft und validiert wird, bevor er in die Produktionsumgebung übergeht.

## Testprotokoll

**Projekt**: Multiuser Casino Web Applikation

**Testumgebung**:

* **Betriebssystem**: Windows 11
* **Browser**: Edge
* **Testwerkzeuge**:
  + Jasmine: 3.9.0
  + Karma: 6.3.2
* **Datenbank**: MongoDB 4.4.1
* Node.js: 14.17.0
* Angular: 17.0.0
* **Tester**: KAND

### Whitebox-Tests (Jasmine):

#### Registrierungstest

* Beschreibung: Testet, ob ein neuer Benutzer mit gültigen Daten erstellt werden kann.
* Testfall:
  + Eingabe: Valide Benutzerdaten (Name, Nachname, Nutzername, Passwort, Avatar, Bank IBAN).
  + Erwartetes Ergebnis: Benutzer wird erfolgreich erstellt und in der Datenbank gespeichert.

#### 2. Login-Test

* Beschreibung: Testet, ob das Login mit korrektem Nutzernamen und Passwort funktioniert.
* Testfall:
  + Eingabe: Gültiger Nutzername und Passwort.
  + Erwartetes Ergebnis: Login wird erfolgreich durchgeführt, Benutzersitzung beginnt.

#### 3. Kontostand-Aktualisierungstest

* Beschreibung: Testet, ob der Kontostand nach einem Slotmaschinenspiel korrekt aktualisiert wird.
* Testfall:
  + Eingabe: Ergebnis eines Slotmaschinenspiels (Gewinn oder Verlust).
  + Erwartetes Ergebnis: Kontostand wird korrekt erhöht oder reduziert.

### Unit-Tests (Karma):

#### 1. Slotmaschinen-Spin-Test

* Beschreibung: Testet, ob ein Spin der Slotmaschine die erwarteten Ergebnisse liefert.
* Testfall:
  + Eingabe: Benutzer tätigt einen Spin mit einem Einsatz.
  + Erwartetes Ergebnis: Ergebnis des Spins wird angezeigt und der Einsatz wird vom Kontostand abgezogen.

#### 2. Blackjack-Spiellogik-Test

* Beschreibung: Testet die Spiellogik von Blackjack gegen die Bank.
* Testfall:
  + Eingabe: Benutzer spielt eine Hand Blackjack.
  + Erwartetes Ergebnis: Spiel wird nach den Blackjack-Regeln korrekt durchgeführt.

#### 3. Benutzerdaten-Validierungstest

* Beschreibung: Testet die Validierung von Benutzerdaten bei der Registrierung.
* Testfall:
  + Eingabe: Benutzerdaten mit absichtlichen Fehlern (z.B. ungültige IBAN).
  + Erwartetes Ergebnis: System erkennt die Fehler und verweigert die Registrierung.

## Dokumentation:

Jeder Testfall wird nach Durchführung dokumentiert und enthält Informationen über den Teststatus (Bestanden/Gescheitert), aufgetretene Fehler und zusätzliche Kommentare des Testers.

Die Testergebnisse werden in der Dokumentation schriftlich festgehalten und sind für den Experten zur Überprüfung und Fehlerbehebung verfügbar.