



Dokumentation GuessIt

Sven Leutenegger | SE2 | 28.05.2023

Inhaltsverzeichnis

Einleitung	3
Die Projektidee.....	3
Problemstellung	3
Benutzergruppen	3
Explore-, Create- und Evaluate-Board.....	4
Erkenntnisse aus dem Pitch	6
Anforderungen	7
Use-Case Diagramm.....	7
Use Cases:.....	7
Use Case Descriptions:.....	8
Fachliches Datenmodell (ER-Modell) mit Erläuterungen	12
Prozessmodell (BPMN-Diagramm) mit Erläuterungen.....	13
Mockup oder Skizze des UIs	13
Implementation	15
Frontend.....	15
Klassendiagramm	19
Beschreibung der DTOs und DAOs	19
Aufgaben und Funktionen eingebundener Drittsysteme	20
Testing.....	20
Modultests	20
AnswerQuestionServiceTest	21
ScoreCalculationTest	22
Integrationtests	24
EndpointTests	24
TestSecurityConfig	26
User-Tests.....	27
Resultate der User-Tests, Diskussion der Ergebnisse.....	30
Codeanalyse mit SonarCube	31
Bugs.....	32
Code Smells.....	32
Fazit	34

Stand der Implementation.....	34
Persönliches Fazit	34

Einleitung

DIE PROJEKTIDEE

GuessIt ist ein Schätzfragespiel, das Spieler dazu herausfordert, ihr Wissen und ihre Schätzfähigkeiten auf die Probe zu stellen. Dieses einzigartige Spielkonzept kombiniert unterhaltsame Räselelemente mit einem spannenden Wettbewerb, indem den Spielern knifflige Schätzfragen gestellt werden. Die Aufgabe besteht darin, diese Fragen innerhalb einer vorgegebenen Zeit von einer Minute zu beantworten.

Die Mechanik von GuessIt ist so gestaltet, dass der Spieler für seine Genauigkeit belohnt wird. Je näher die Schätzung an der richtigen Antwort liegt, desto mehr Punkte werden dem Spieler gutgeschrieben. Diese Punkte sind nicht nur ein Indikator für den Erfolg des Spielers, sondern auch ein Weg, um im Spiel Fortschritte zu machen und sich in den Ranglisten zu verbessern.

Das ultimative Ziel für jeden Spieler in GuessIt ist es, das letzte Level zu vollenden und einen Spitzenplatz im Leaderboard zu erlangen. Dieses Leaderboard ist ein wesentlicher Bestandteil von GuessIt, da es einen gesunden Wettbewerb zwischen den Spielern fördert und jedem die Möglichkeit gibt, seine Fähigkeiten und sein Wissen im Vergleich zu anderen Spielern unter Beweis zu stellen.

Problemstellung

GuessIt stellt sich der Herausforderung, Menschen einen ansprechenden und interaktiven Weg zu bieten, ihr Wissen zu erweitern und gleichzeitig ihre kognitiven Fähigkeiten zu schärfen. In einer Welt, die immer mehr auf Informationsaustausch und Lernen durch digitale Plattformen setzt, bietet GuessIt eine Alternative zur traditionellen Wissensvermittlung.

Das Spiel ist nicht nur ein unterhaltsamer Zeitvertreib, sondern hilft auch dabei, die Fähigkeiten in Bezug auf kritisches Denken, Problemlösung und Entscheidungsfindung zu verbessern. Darüber hinaus fördert es die Konzentration, da es die Spieler dazu ermutigt, genaue Schätzungen abzugeben und dabei die verfügbare Zeit effizient zu nutzen.

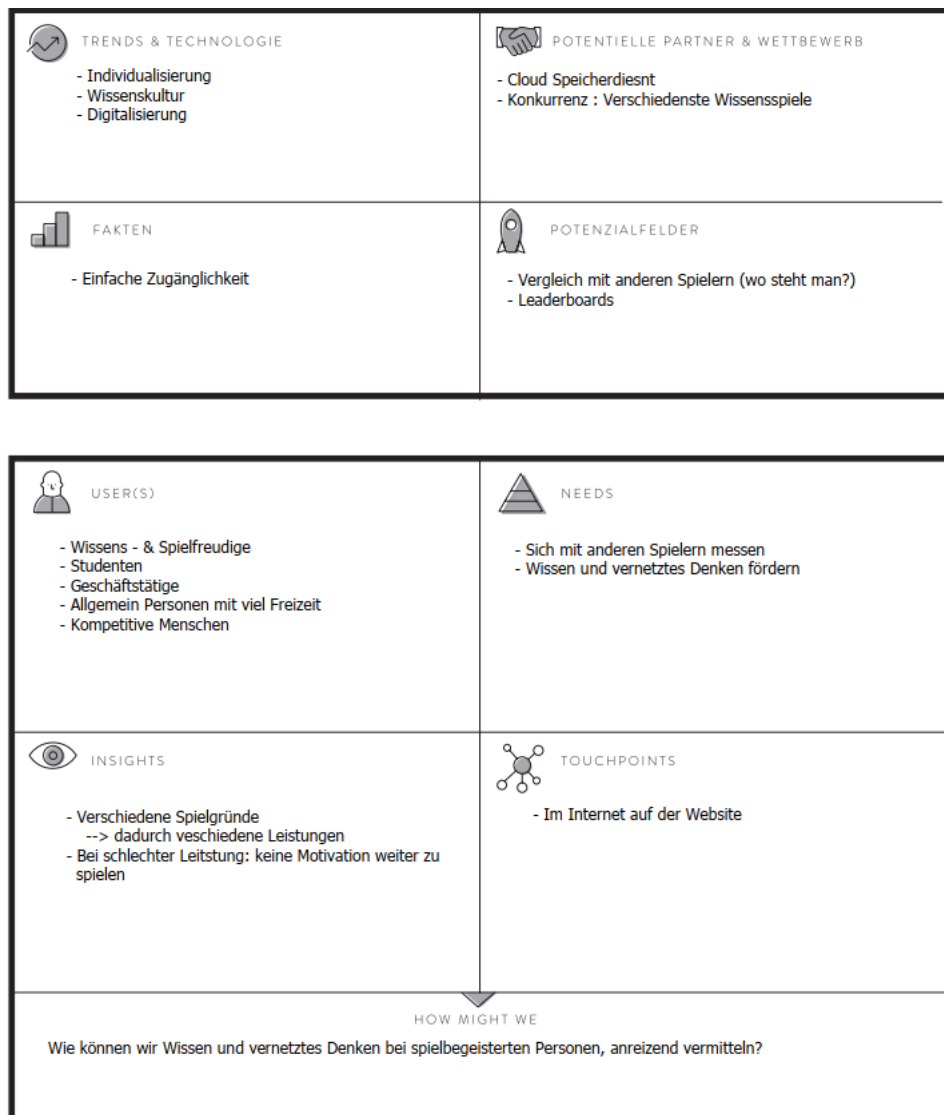
Benutzergruppen

GuessIt spricht eine breite Palette von Benutzergruppen an. Es ist ein ideales Spiel für alle, die ihre allgemeinen Kenntnisse und Schätzfähigkeiten verbessern möchten, sei es für Studierende, die ihr Wissen auf die Probe stellen wollen, Berufstätige, die ihre Denkfähigkeiten schärfen möchten, oder für Senioren, die ihren Geist aktiv halten möchten.

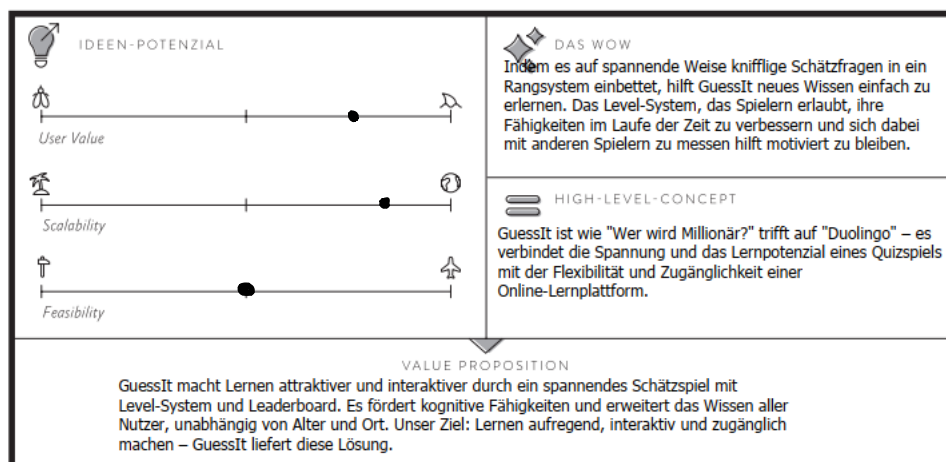
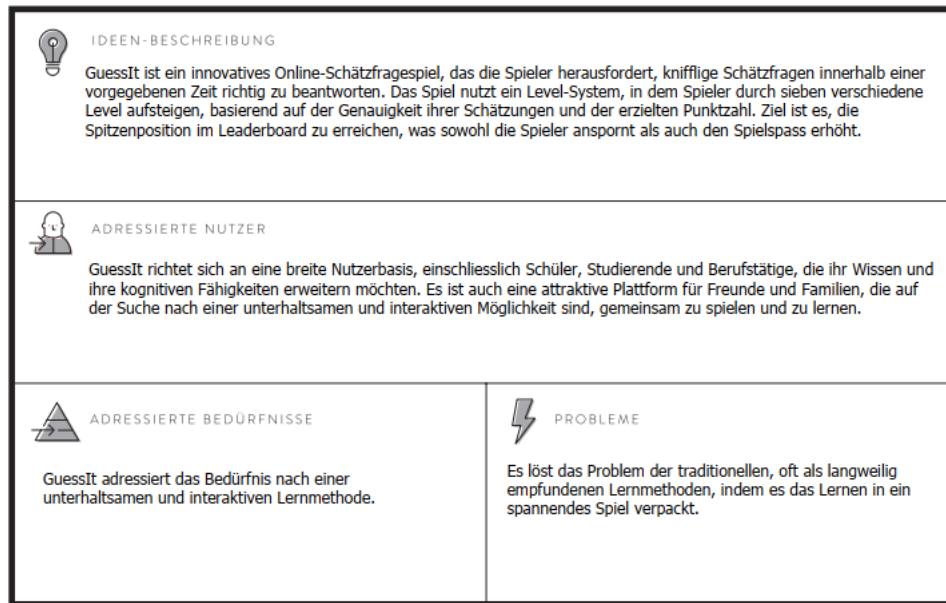
GuessIt ist also mehr als nur ein Spiel. Es ermöglicht den Nutzern, ihr Wissen auf eine unterhaltsame und interaktive Weise zu erweitern, ihre geistigen Fähigkeiten zu verbessern und gleichzeitig mit anderen in einem gesunden Wettbewerb zu interagieren.

EXPLORE-, CREATE- UND EVALUATE-BOARD











Explore Board:



Create Board:



Evaluate Board:

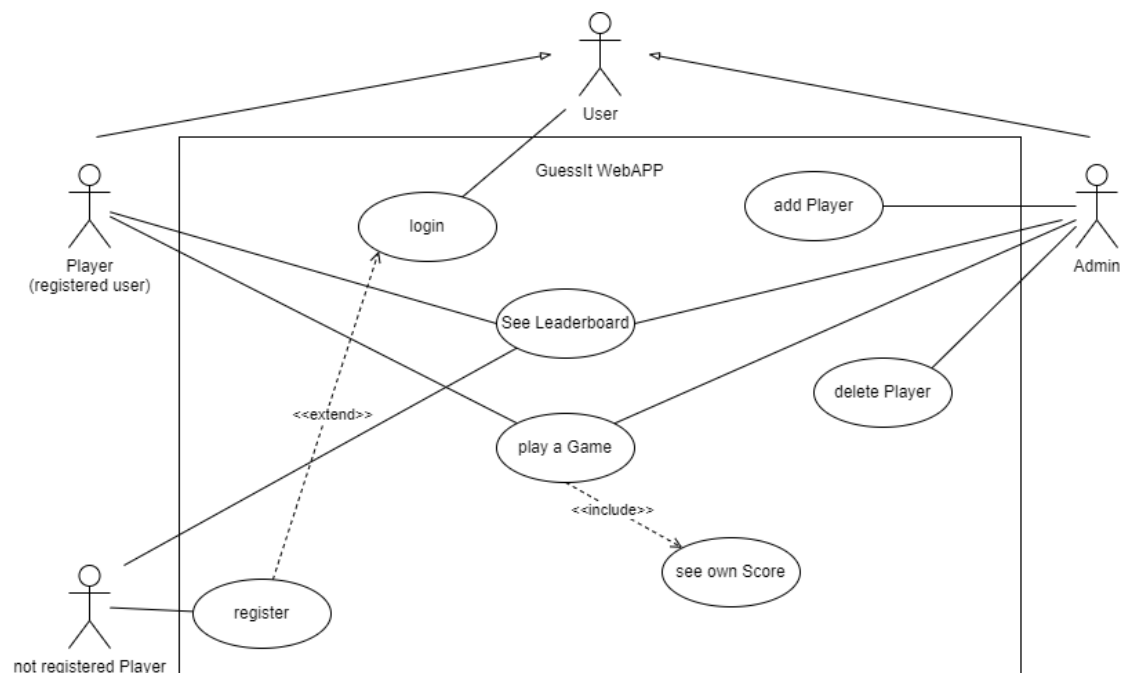
<p> Wertversprechen-Skala</p> <p>0 10</p> <p>User is activated</p> <p>0 10</p> <p>User prefers prototype to similar product</p> <p>0 10</p> <p>User would buy it</p> <p>0 10</p> <p>User would recommend it</p>	<p> Preispunkt & Kaufbereitschaft</p> <p>Für weitere Levels: 1-3 CHF</p> <hr/> <p> User Experience</p> <p> </p> <p>Emotion</p>
<p> Assessment</p> <p>Brand Fit</p> <p>Market Size</p> <p>Investment</p> <p>Asset Fit</p> <p>Viral Potential</p> <p>New Customers</p>	<p> Kanäle</p> <p>Social Media, Word of Mouth Marketing</p> <hr/> <p> Unfairer Vorteil</p> <p>Kreative Schätzfragen</p> <hr/> <p> KPI</p> <p>Anzahl User, Anzahl beantworteten Fragen</p> <hr/> <p> Einnahme-Quellen</p> <p>Kostenpflichtige Spielupdates z.B. neue Levels.</p> <p>Kostenpflichtiger Re-Run (neue Fragen)</p>

ERKENNTNISSE AUS DEM PITCH

Die positive Resonanz auf den Pitch meiner Projektidee Guesst, hat die Vielseitigkeit und Attraktivität des Konzepts bestätigt. Es war aufschlussreich zu hören, dass das Spielkonzept Interesse bei einer breiten Zielgruppe wecken konnte - ein Zeichen dafür, dass die Spielidee universell ansprechend sein könnte. Weder die Mitstudierenden noch der Dozent äusserten Kritikpunkte, was für mich ein starkes Indiz dafür war, dass die Idee kreativ, praktikabel und auch gut realisierbar ist. Diese durchweg positive Rückmeldung motivierten mich, ein hochwertiger und ansprechender Prototyp zu entwickeln.

Anforderungen

USE-CASE DIAGRAMM



USE CASES:

Use Case	ID	
Register	1	Ein neuer Benutzer besucht die GuessIt-Website. Er klickt auf die Schaltfläche "Registrieren" und füllt die erforderlichen Felder aus, darunter seine E-Mail-Adresse und sein Passwort. Nach dem Absenden des Formulars validiert das System die eingegebenen Daten und erstellt ein neuer Benutzer.
Login	2	Ein registrierter Benutzer ruft die GuessIt-Website auf und klickt auf die Schaltfläche "Login". Er gibt seine registrierte E-Mail-Adresse und sein Passwort ein. Nach erfolgreicher Überprüfung der eingegebenen Daten meldet das System den Benutzer an und gewährt ihm Zugang zu den Spielfunktionen.
See leaderboards	3	Ein Benutzer möchte die besten Spieler überprüfen. Er navigiert zum Abschnitt "Leaderboards", wo er die Punktzahlen und Ranglisten der besten Spieler sehen kann, was den Wettbewerbsgeist anregt.
Play a game	4	Ein registrierter Benutzer möchte ein Spiel spielen. Er wählt den Button "Play", wo ihm dann eine Reihe von Fragen gestellt wird. Er gibt seine Antworten innerhalb des vorgegebenen Zeitrahmens ab.
See own Score	5	Nach Beendigung jeder Frage wird der Punktestand des Nutzers berechnet und aktualisiert angezeigt.
Add Player	6	Ein Administrator möchte einen neuen Spieler manuell hinzufügen. Er navigiert zum Abschnitt "Players", füllt die erforderlichen Felder aus und sendet das Formular ab. Das

		System validiert die eingegebenen Daten und erstellt ein neues Spielerprofil.
Delete Player	7	Ein Administrator muss das Konto eines Spielers löschen. Er navigiert zum Abschnitt "Player", sucht das Profil des betreffenden Spielers und klickt auf die Schaltfläche "delete". Nachdem auf die Schaltfläche gedrückt hat, löscht das System das Konto des Spielers aus der Datenbank.

USE CASE DESCRIPTIONS:

Titel	Register
ID	1
Actors	Unregistrierter Benutzer
Eintrittsbedingungen	Dieser Anwendungsfall beginnt, wenn ein neuer Benutzer die Guesst-Website besucht und beschliesst, ein neues Konto anzulegen.
Ereignissequenz	Der Nutzer klickt auf die Schaltfläche "Registrieren", füllt die erforderlichen Felder aus (einschliesslich E-Mail und Passwort) und sendet das Formular ab. Das System validiert die eingegebenen Daten und erstellt ein neues Benutzerkonto.
Austrittsbedingungen	Dieser Anwendungsfall endet, wenn das neue Benutzerkonto erfolgreich erstellt wurde.
Ausnahmen	Wenn die eingegebenen Daten ungültig sind (z. B., weil die E-Mail bereits verwendet wird), zeigt das System eine Fehlermeldung an.
Besondere Anforderungen	Der Registrierungsprozess muss den Datenschutzbestimmungen entsprechen.
Daten	Der Nutzer muss eine E-Mail-Adresse und ein Passwort angeben.

Titel	Login
ID	2
Actors	Registrierter Benutzer
Eintrittsbedingungen	Dieser Anwendungsfall beginnt, wenn ein registrierter Benutzer die Guesst-Website besucht und sich anmelden möchte.

Ereignissequenz	Der Benutzer klickt auf die Schaltfläche "Anmelden", gibt seine registrierte E-Mail-Adresse und sein Passwort ein und sendet das Formular ab. Das System überprüft die eingegebenen Anmeldedaten.
Austrittsbedingungen	Dieser Anwendungsfall endet, wenn der Benutzer erfolgreich eingeloggt ist.
Ausnahmen	Wenn die eingegebenen Anmeldedaten falsch sind, zeigt das System eine Fehlermeldung an.
Besondere Anforderungen	Das Anmeldeverfahren muss sicher sein, um unbefugten Zugriff zu verhindern.
Daten	Der Benutzer muss seine registrierte E-Mail-Adresse und sein Passwort eingeben

Titel	See leaderboards
ID	3
Actors	Alle Benutzer
Eintrittsbedingungen	Dieser Anwendungsfall beginnt, wenn ein Benutzer zum Bereich "Leaderboard" navigiert.
Ereignissequenz	Der Benutzer betrachtet die Rangliste, die die Top-Spieler und ihre Scores anzeigt.
Austrittsbedingungen	Dieser Anwendungsfall endet, wenn der Benutzer die Seite "Leaderboard" verlässt.
Ausnahmen	Wenn die Rangliste nicht geladen werden kann, zeigt das System eine Fehlermeldung an
Besondere Anforderungen	Die Rangliste muss in Echtzeit aktualisiert werden, um die Platzierungen korrekt wiederzugeben.
Daten	Die Rangliste zeigt Spielerplatzierungen und -punkte an.

Titel	Play a game
ID	4
Actors	Registrierter Benutzer, Admin

Eintrittsbedingungen	Dieser Anwendungsfall beginnt, wenn ein registrierter Benutzer oder ein Admin auf den Button "Play" klickt.
Ereignissequenz	Das System präsentiert eine Reihe von Fragen, die innerhalb eines bestimmten Zeitrahmens beantwortet werden müssen.
Austrittsbedingungen	Dieser Anwendungsfall endet, wenn das Spiel abgeschlossen ist oder der Benutzer beschliesst, nicht weiter zu spielen.
Ausnahmen	Bei technischen Problemen während des Spiels zeigt das System eine Fehlermeldung an.
Besondere Anforderungen	Die Spielschnittstelle muss benutzerfreundlich sein und die Fragen müssen fair und unvoreingenommen sein.
Daten	Das System speichert die Antworten des Benutzers und die sich daraus ergebenden Punkte.

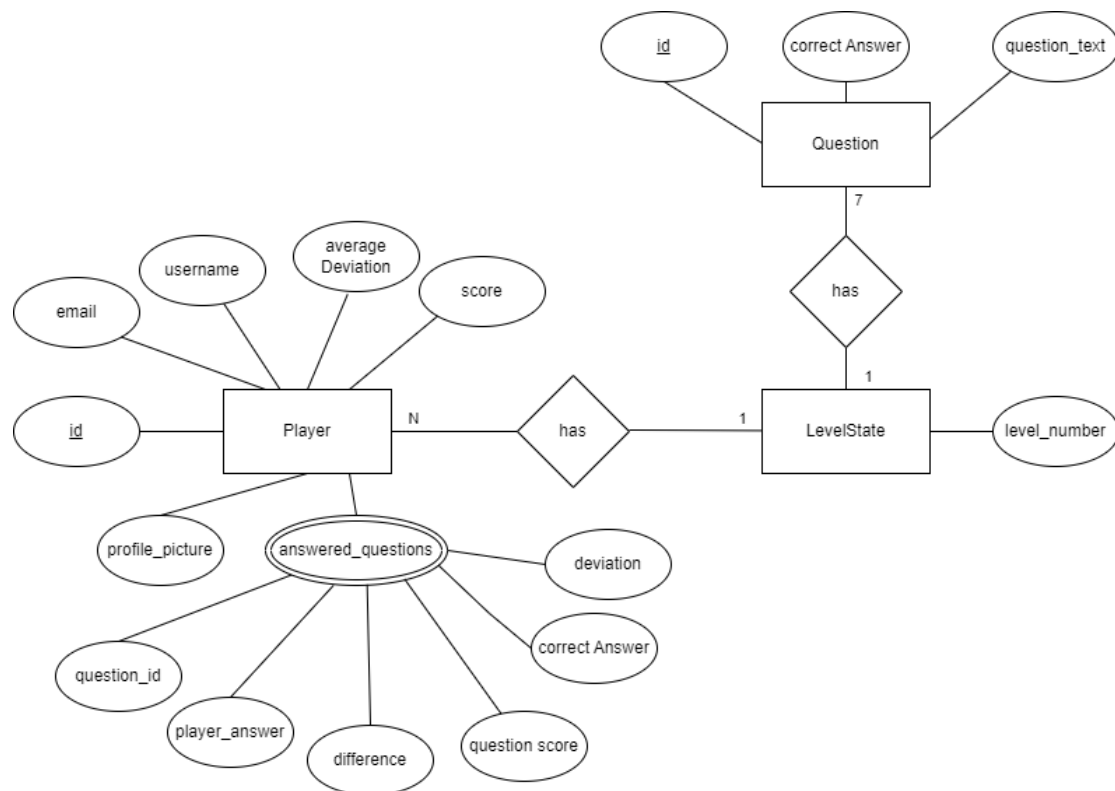
Titel	See own Score
ID	5
Actors	Registrierter Benutzer, Admin
Eintrittsbedingungen	Dieser Anwendungsfall beginnt, wenn ein Benutzer oder Admin eine Frage beantwortet hat.
Ereignissequenz	Das System berechnet und zeigt die aktualisierte Punktzahl des Benutzers an.
Austrittsbedingungen	Dieser Anwendungsfall endet, wenn der Benutzer die Anzeige der Punktzahl durch Klick auf Next verlässt.
Ausnahmen	Wenn die Punktzahl nicht geladen oder aktualisiert wird, zeigt das System eine Fehlermeldung an
Besondere Anforderungen	Der Punktberechnungsalgorithmus muss genau und fair sein.
Daten	Das System zeigt die aktualisierte Punktzahl des Benutzers basierend auf seiner Leistung im Spiel an.

Titel	Add Player
ID	6
Actors	Admin
Eintrittsbedingungen	Dieser Anwendungsfall beginnt, wenn ein Administrator einen neuen Spieler manuell hinzufügen möchte.
Ereignissequenz	Der Administrator navigiert zum Abschnitt "Player", füllt die erforderlichen Felder aus und sendet das Formular ab. Das System validiert die eingegebenen Daten und erstellt ein neues Spielerprofil.
Austrittsbedingungen	Dieser Anwendungsfall endet, wenn das Profil des neuen Spielers erfolgreich erstellt wurde.
Ausnahmen	Wenn die eingegebenen Daten ungültig sind oder ein technisches Problem auftritt, zeigt das System eine Fehlermeldung an
Besondere Anforderungen	Der Administrator muss die entsprechenden Berechtigungen haben, um Spieler hinzufügen zu können.
Daten	Das System verarbeitet die Daten des neuen Spielers, die während dieses Prozesses eingegeben werden. Diese Daten beinhalten den Spielernamen und die E-Mail-Adresse des Spielers.

Titel	Delete Player
ID	7
Actors	Admin
Eintrittsbedingungen	Dieser Anwendungsfall beginnt, wenn ein Administrator das Konto eines Spielers löschen muss.
Ereignissequenz	Der Administrator navigiert zum Abschnitt "Player", sucht das Profil des betreffenden Spielers und klickt auf die Schaltfläche "Delete". Das System löscht das Konto des Spielers aus der Datenbank.
Austrittsbedingungen	Dieser Anwendungsfall endet, wenn das Konto des Spielers erfolgreich gelöscht wurde

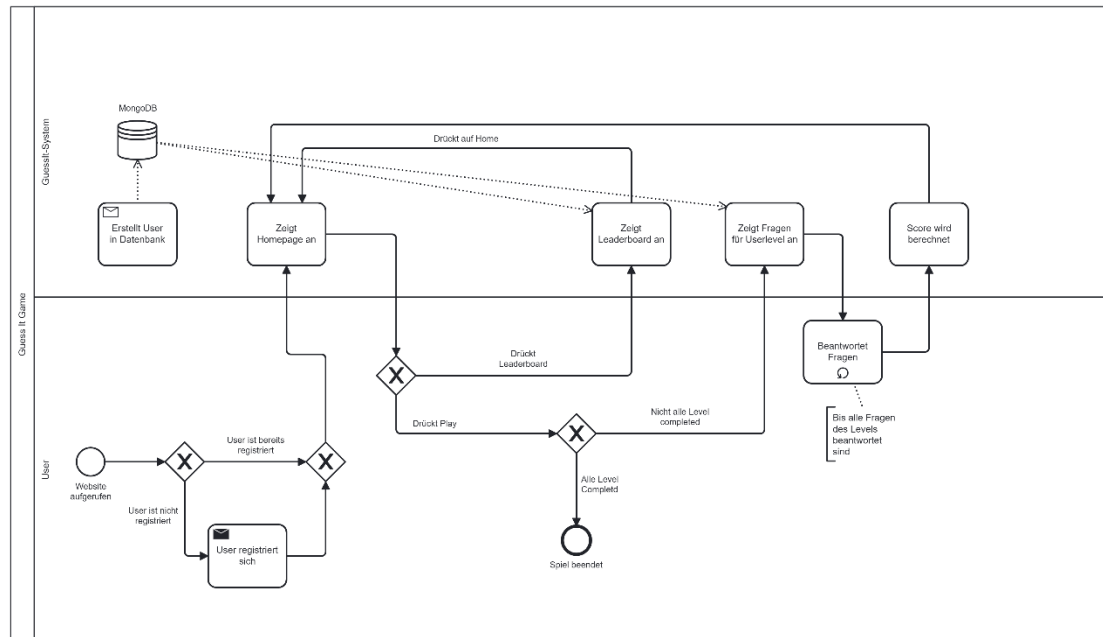
Ausnahmen	Wenn es während des Löschvorgangs zu technischen Problemen kommt, zeigt das System eine Fehlermeldung an.
Besondere Anforderungen	Der Administrator muss die entsprechenden Berechtigungen haben, um Spieler zu löschen.
Daten	Das System verarbeitet die Kontodaten des Spielers während dieses Prozesses. Die Daten die Id des Spielers.

FACHLICHES DATENMODELL (ER-MODELL)



Es gibt zwei Collections in der Datenbank, Player und Question. Jeder Player hat eine Id, eine Mailadresse, ein Username, eine durchschnittliche Abweichung, ein Score und eine Liste von beantworteten Fragen. Eine Frage hat ein Fragetext, die korrekte Antwort und eine Id. Verbunden sind beide über das Attribut LevelState. Dieses kann sich beim Spieler ändern, jede Question hat jedoch ein fixes Level. Das LevelState ist die Verbindung um dann zu jedem Spieler die Fragen anzuzeigen auf dessen Level er ist.

PROZESSMODELL (BPMN-DIAGRAMM) MIT ERLÄUTERUNGEN




Die BPMN zeigt den Prozess wie das Spiel durchgespielt werden kann.

1. Der Prozess beginnt, wenn ein User die Website aufruft.
2. Es wird geprüft, ob der Benutzer bereits registriert ist. Es gibt zwei mögliche Wege, abhängig vom Ergebnis dieser Prüfung:
 - Wenn der Benutzer nicht registriert ist, muss er sich registrieren. Der Benutzer wird in der Datenbank erstellt.
 - Wenn der Benutzer bereits registriert ist, wird dieser Pfad verfolgt.
3. Nachdem der Benutzer registriert ist, wird er auf die Homepage weitergeleitet.
4. Auf der Homepage kann der Benutzer entweder auf "Play" drücken oder das Leaderboard betrachten.
5. Wenn der Benutzer auf Leaderboard klickt, wird das Leaderboard angezeigt.
6. Wenn der Benutzer "Play" drückt, wird geprüft, ob er alle Level abgeschlossen hat.
 - Wenn nicht alle Level abgeschlossen sind, werden Fragen für das entsprechende Benutzerlevel angezeigt. Der Benutzer beantwortet Fragen und der Punktestand wird berechnet.
 - Wenn alle Level abgeschlossen sind, ist das Spiel beendet. Dies ist das Endereignis.

MOCKUP ODER SKIZZE DES UIS

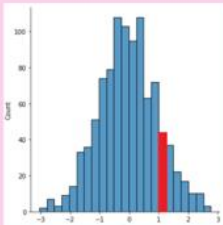
Folgende Mockups wurden für den Pitch in SW 3 erstellt:

Homepage:



Username









Your Performance:



GUESS IT

PLAY

leaderboard:

1			Player Name	★★★★★	2980
2			Player Name	★★★★☆	2721
3			Player Name	★★★★☆	2579
4			Player Name	★★★★☆	1874
5			Player Name	★★★★☆	1756

coming soon

LEVEL 7

LEVEL 6

LEVEL 5

LEVEL 4

LEVEL 3

LEVEL 2

LEVEL 1

Play:

Question: How many keys are on a computerkeyboard on average?

Your answer:

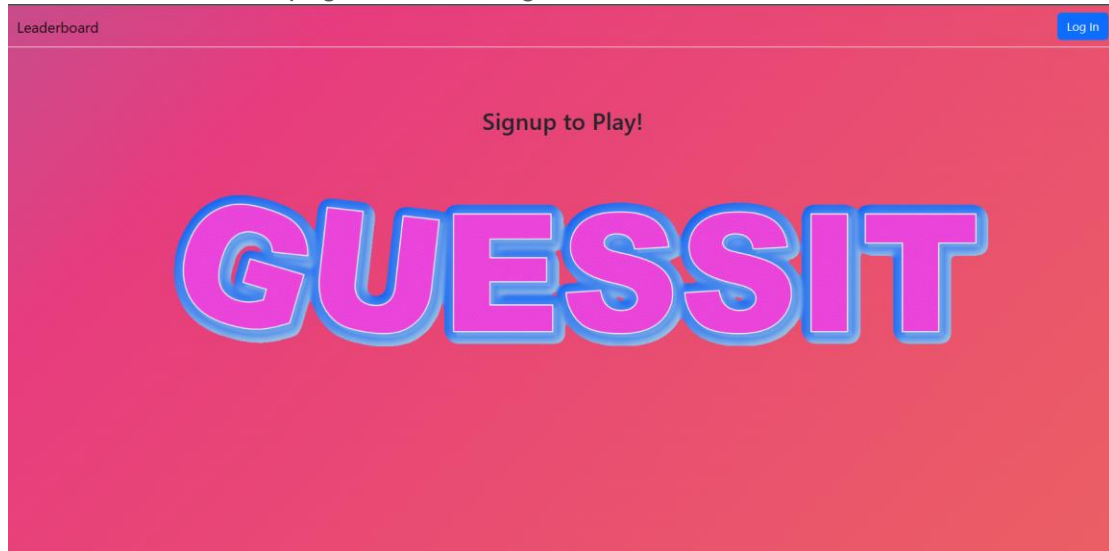
guess it!

Correct answer: **104**

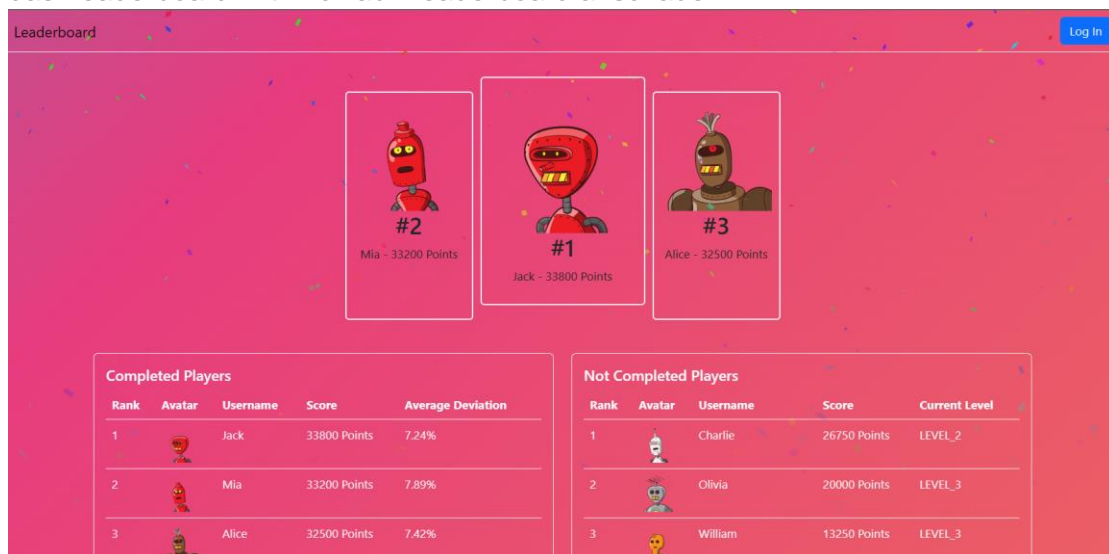
Implementation

FRONTEND

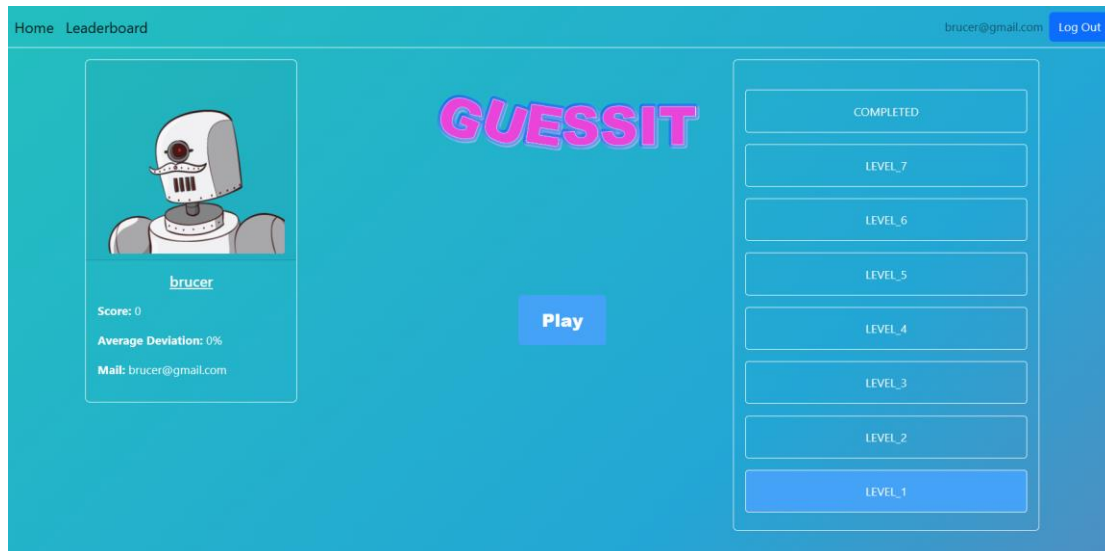
Bei Aufrufen der Webpage erscheint folgendes:



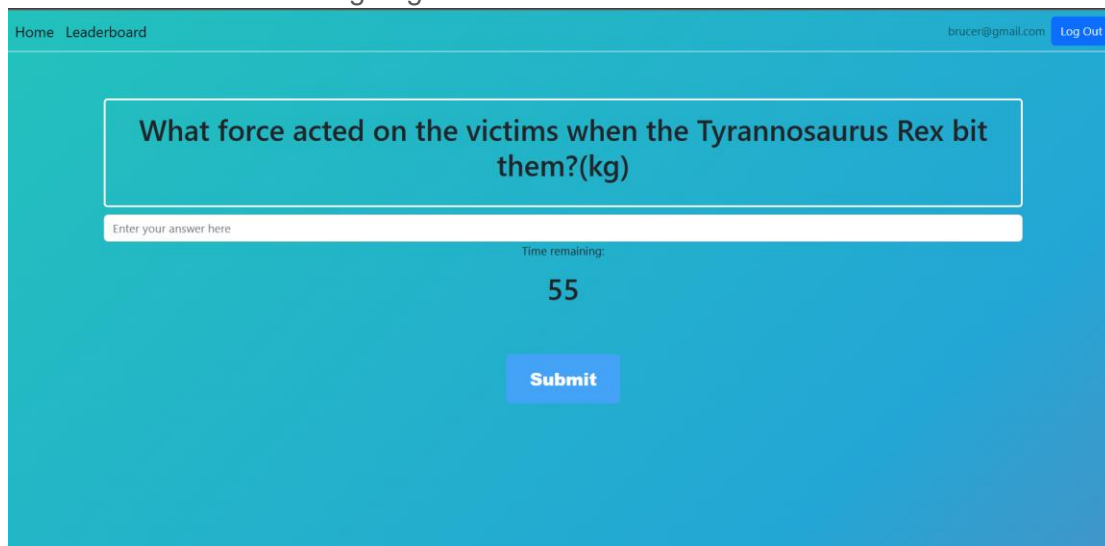
Als nicht registrierte User hat man keine Möglichkeit zu spielen, allerdings kann man das Leaderboard mit Klick auf Leaderboard anschauen.



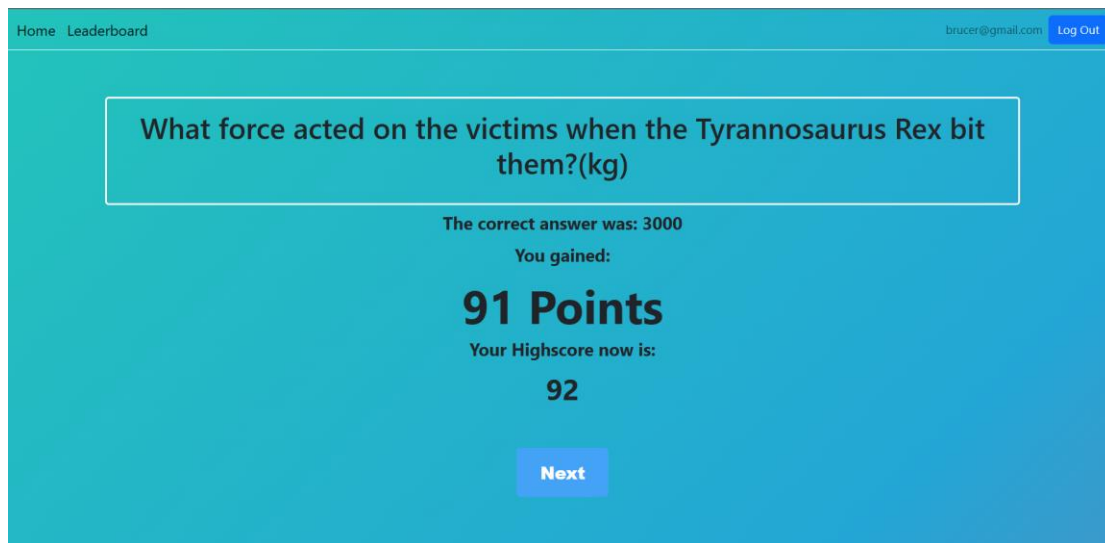
Möchte man Spielen muss man sich einloggen, oder falls man noch nicht registriert ist, registrieren. Nach dem Login wird man auf die Startseite weitergeleitet.



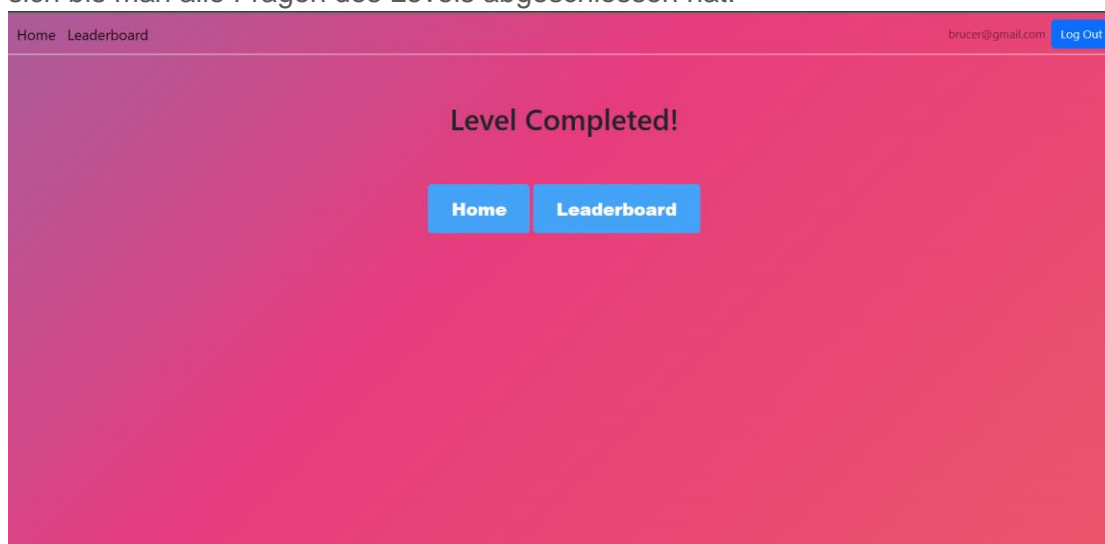
Möchte man nun spielen, klickt man auf Play und muss dann innerhalb von 60 Sekunden seine Schätzung abgeben.



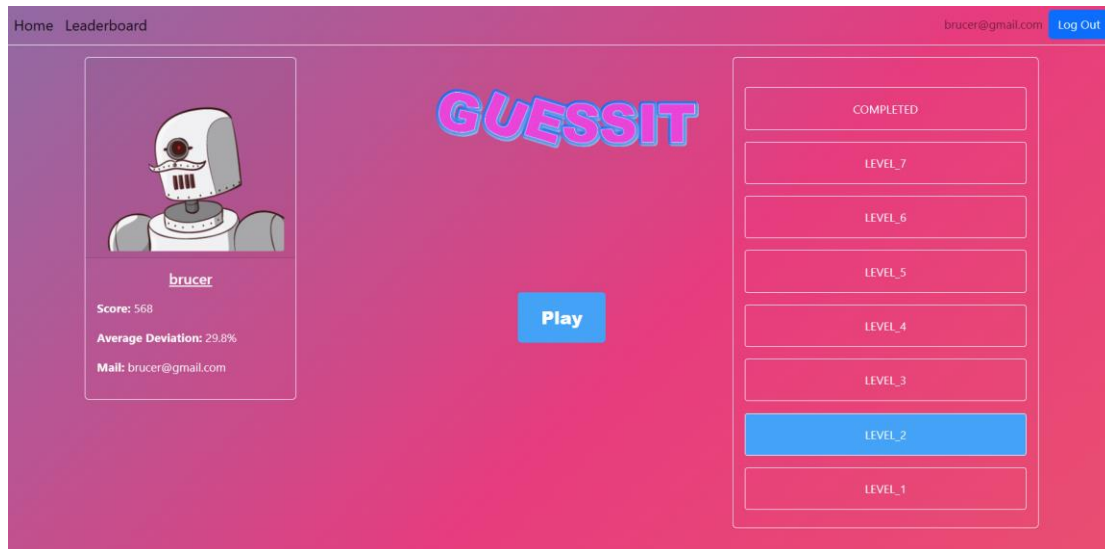
Nachdem man seine Schätzung submitted hat wird einem die richtige Antwort und die erhaltenen Punkte für diese Frage, sowie deine Highscore angezeigt.



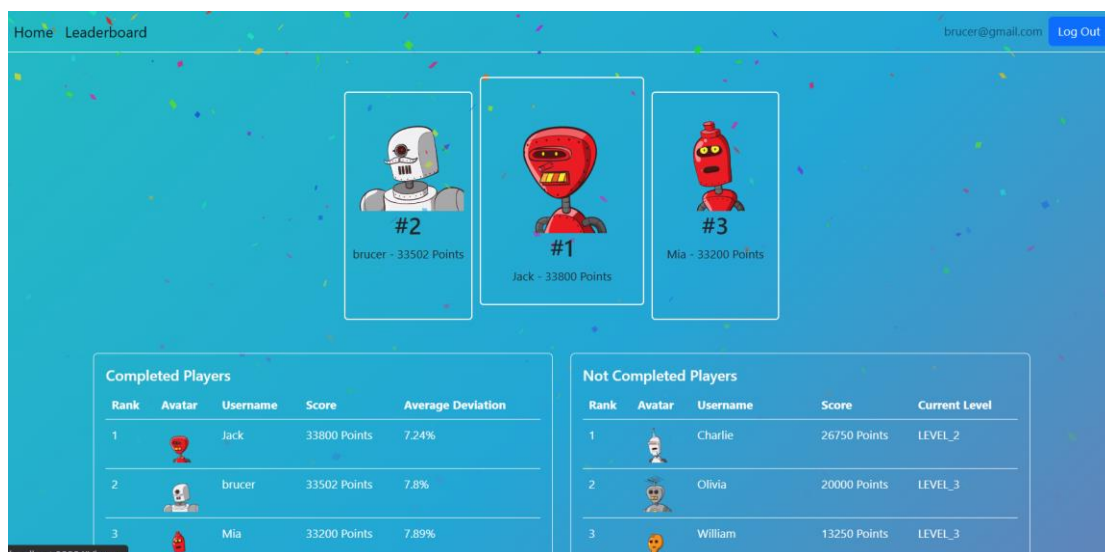
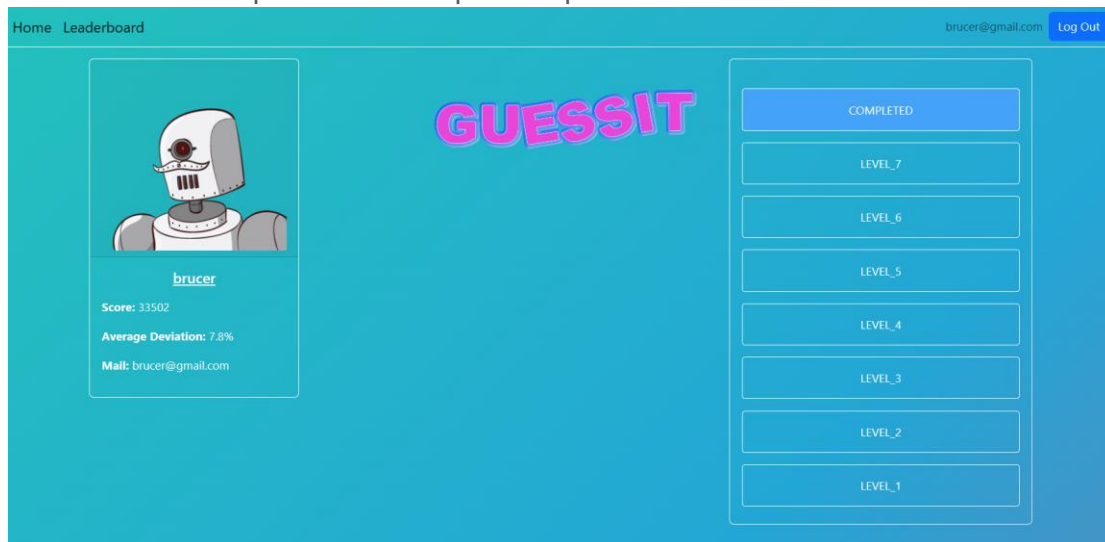
Um weiterzuspielen klickt man auf Next. Un erhält die nächste Frage. Dies wiederholt sich bis man alle Fragen des Levels abgeschlossen hat.



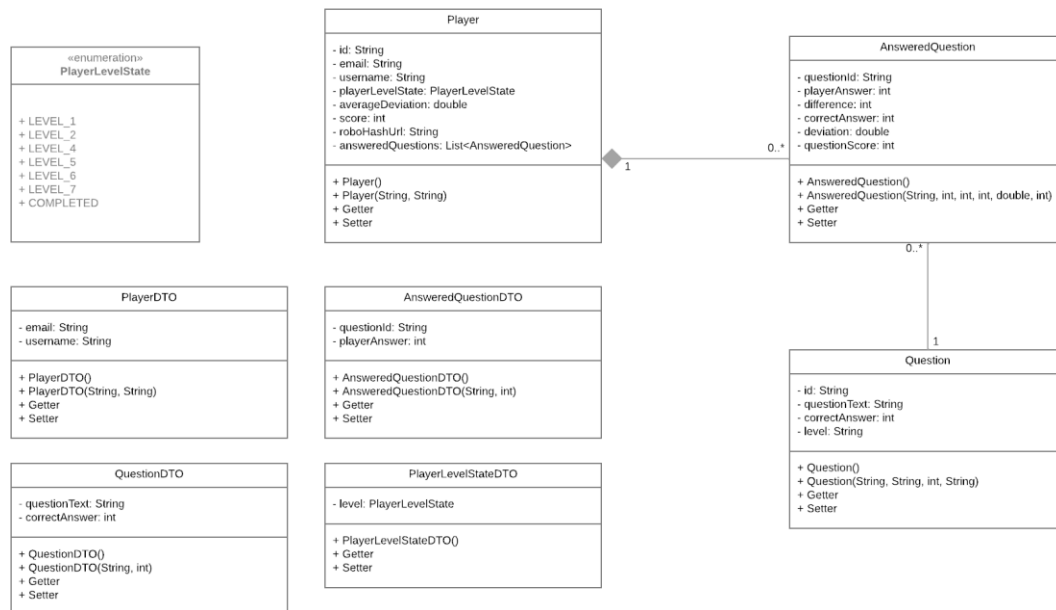
Danach hat man die Möglichkeit gleich das Leaderboard und seinen Stand anzusehen, oder zurück zur Startseite zu gelangen. Wenn wir nun zurück zu Startseite gehen, haben sich nun unsere Stats und das Level angepasst.



Beenden wir alle Levels verschwindet der Play Button und wir kommen ins Leaderboard der Spieler die das Spiel completed haben.



KLASSENDIAGRAMM



Beschreibung der DTOs und DAOs

DTOs (Data Transfer Objects):

AnsweredQuestionDTO: Dieses DTO enthält Informationen zu einer vom Spieler beantworteten Frage. Es beinhaltet die ID der Frage und der vom Spieler gegebenen Antwort. Es wird verwendet, wenn ein Spieler eine Frage beantwortet, um die Daten vom Client an den Server zu übertragen.

PlayerDTO: Dieses DTO enthält die E-Mail und Benutzername zu einem Spieler. Es wird verwendet, um Spielerdaten vom Client an den Server zu übertragen, z.B. nach der Registrierung eines neuen Spielers.

QuestionDTO: Dieses DTO enthält Informationen zu einer Frage, einschliesslich des Fragetextes und der richtigen Antwort. Es wird verwendet, um Frage-Daten vom Client an den Server zu übertragen.

PlayerLevelStateDTO: Dieses DTO enthält Informationen über den aktuellen Level eines Spielers. Es wird verwendet, um Daten über den Level eines Spielers vom Client an den Server zu übertragen. Damit wird nach beenden das Level des Spielers geupdated.

DAOs (Data Access Objects):

PlayerRepository: Dieses Repository bietet Methoden zur Interaktion mit der Datenquelle für Spielerdaten. Es wird zum Abrufen über E-Mail, Speichern, Aktualisieren und Löschen von Spielerdaten verwendet.

QuestionRepository: Dieses Repository bietet Methoden zur Interaktion mit der Datenquelle für Frage-Daten. Es wird zum Abrufen von Frage-Daten verwendet.

AUFGABEN UND FUNKTIONEN EINGEBUNDENER DRITTSYSTEME

RoboHash:

Die RoboHash API ist ein Drittsystem, das in der Anwendung genutzt wird, um ein einzigartiges und visuell ansprechendes Profilbild für jeden Spieler zu generieren. Bei der Registrierung oder der Erstellung eines neuen Spielers wird die RoboHash API aufgerufen, um ein zufälliges, roboterähnliches Bild basierend auf dem Benutzernamen des Spielers zu erstellen.

Das Bild wird dann über eine URL zugänglich gemacht, die in der Player-Klasse als "roboHashUrl" gespeichert wird. Diese URL führt direkt zum generierten Bild auf den RoboHash-Servern.

Der Prozess des Generierens und Speicherns der RoboHash-URL wird in der "setUsername"-Methode der Player-Klasse ausgeführt. Wenn der Benutzername eines Spielers festgelegt oder geändert wird, wird auch die "roboHashUrl" aktualisiert, um ein neues Bild zu reflektieren.

```
public void setUsername(String username) {  
    this.username = username;  
    this.roboHashUrl = "https://robohash.org/" + username + ".png";  
}
```

Diese Integration mit dem RoboHash-API ermöglicht es, jedem Spieler in der Anwendung ein einzigartiges und attraktives Profilbild zuzuweisen, ohne dass wir uns um die Speicherung oder Generierung der Bilder kümmern müssen. Darüber hinaus bietet sie den Spielern eine visuelle Identität innerhalb des Spiels, was zur Verbesserung des Spielerlebnisses beiträgt.

Testing

MODULTESTS

Modluteests finden sich um die Berechnung der Abweichung zwischen Spielerantwort und richtiger Antwort (answerQuestionTest), die Berechnung der Abweichung (answerQuestionDeviationTest) und die Durchschnittsberechnung der Abweichung (averageDeviationTest) zu überprüfen. Zudem wird in ScoreCalculationTest die korrekte Berechnung der Gesamt- und Einzelpunktzahlen überprüft. Diese Tests nutzen vordefinierte Testdaten, welche von der Methode provideTestData bzw. provideDeviationTestData oder provideAverageDeviationTestData geliefert werden.

Für diese Tests werden die Abhängigkeiten zu den Repositories mit Hilfe von Mock-Objekten simuliert, sodass die Tests unabhängig von der Datenbank durchgeführt werden können.

AnswerQuestionServiceTest

Dieser Modultest zielt darauf ab, den AnswerQuestionService zu testen, eine Dienstklasse, die die Logik für das Beantworten von Fragen in der Anwendung enthält. Der Test simuliert verschiedene Szenarien, um sicherzustellen, dass der Service korrekt funktioniert.

setUp(): Diese Methode wird vor jedem Test ausgeführt, um die Testdaten vorzubereiten und das Verhalten der Mock-Repositories zu konfigurieren.

answerQuestionTest(): Dieser Test überprüft die Berechnung des Unterschieds zwischen der richtigen Antwort und der vom Spieler gegebenen Antwort.

answerQuestionDeviationTest(): Dieser Test überprüft die Berechnung der Abweichung des Spielers von der korrekten Antwort, gemessen in Prozent.

averageDeviationTest(): Dieser Test überprüft die Berechnung der durchschnittlichen Abweichung des Spielers über mehrere Fragen hinweg.

```
27
28 @ExtendWith(MockitoExtension.class)
29 class AnswerQuestionServiceTest {
30
31     // Inject the mocks into the AnswerQuestionService instance for testing
32     @InjectMocks
33     private AnswerQuestionService answerQuestionService;
34
35     // Mock the PlayerRepository and QuestionRepository dependencies
36     @Mock
37     private PlayerRepository playerRepository;
38
39     @Mock
40     private QuestionRepository questionRepository;
41
42     private Player testPlayer;
43     private Question testQuestion;
44     private static List<AnsweredQuestion> answeredQuestionsAccumulator = new ArrayList<>();
45
46     // Set up the test data before each test method
47     @BeforeEach
48     public void setUp() {
49         testPlayer = new Player(email:"test@example.com", username:"TestUser");
50         testPlayer.setId(id:"1");
51         testPlayer.setAnsweredQuestions(new ArrayList<>());
52
53         testQuestion = new Question(id:"1", questionText:"Test question", correctAnswer:42, level:"LEVEL_1");
54
55         // Configure the mock repositories to return the test data
56         when(playerRepository.findById("1")).thenReturn(Optional.of(testPlayer));
57         when(questionRepository.findById("1")).thenReturn(Optional.of(testQuestion));
58         when(playerRepository.save(any(type:Player.class))).thenAnswer(i -> i.getArguments()[0]);
59
60     }
61
62     // Provide test data for testing the difference calculation
63     private static Stream<Arguments> provideTestData() {
64         return Stream.of(
65             Arguments.of(50, 8),
66             Arguments.of(44, 2),
67             Arguments.of(54, 12));
68     }
69
70     // Parameterized test for testing the difference calculation
71     @ParameterizedTest
72     @MethodSource("provideTestData")
73     void answerQuestionTest(int playerAnswer, int expectedDifference) {
74         AnsweredQuestionDTO answeredQuestionDTO = new AnsweredQuestionDTO(questionId:"1", playerAnswer);
75
76         Player updatedPlayer = answerQuestionService.answerQuestion(playerId:"1", answeredQuestionDTO);
```

```

76     Player updatedPlayer = answerQuestionService.answerQuestion(playerId:"1", answeredQuestionDTO);
77
78     int actualDifference = updatedPlayer.getAnsweredQuestions().get(index:0).getDifference();
79
80     // Assert that the calculated difference matches the expected difference
81     assertEquals(expectedDifference, actualDifference);
82 }
83
84 // Provide test data for testing the deviation calculation
85 private static Stream<Arguments> provideDeviationTestData() {
86     return Stream.of(
87         Arguments.of(50, 19.05),
88         Arguments.of(44, 4.76),
89         Arguments.of(54, 28.57));
90 }
91
92 // Parameterized test for testing the deviation calculation
93 @ParameterizedTest
94 @MethodSource("provideDeviationTestData")
95 void answerQuestionDeviationTest(int playerAnswer, double expectedDeviation) {
96     AnsweredQuestionDTO answeredQuestionDTO = new AnsweredQuestionDTO(questionId:"1", playerAnswer);
97     Player updatedPlayer = answerQuestionService.answerQuestion(playerId:"1", answeredQuestionDTO);
98
99     double actualDeviation = updatedPlayer.getAnsweredQuestions().get(index:0).getDeviation();
100
101     // Assert that the calculated deviation matches the expected deviation within a
102     // tolerance of 0.01 (two decimal places)
103     assertEquals(expectedDeviation, actualDeviation, delta:0.01);
104 }
105
106 // Provide test data for testing the average deviation calculation
107 private static Stream<Arguments> provideAverageDeviationTestData() {
108     return Stream.of(
109         Arguments.of(50, 19.05), // First question, average deviation is the same as the deviation itself
110         Arguments.of(33, 20.24), // Second question, average deviation = (19.05 + 21.43) / 2 = 20.24
111         Arguments.of(59, 26.99) // Third question, average deviation = (19.05 + 21.43 + 40.48) / 3 = 26.99
112     );
113 }
114
115 @ParameterizedTest
116 @MethodSource("provideAverageDeviationTestData")
117 void averageDeviationTest(int playerAnswer, double expectedAverageDeviation) {
118     AnsweredQuestionDTO answeredQuestionDTO = new AnsweredQuestionDTO(questionId:"1", playerAnswer);
119
120     // Use the accumulator to store answered questions before setting them for the test player
121     testPlayer.setAnsweredQuestions(new ArrayList<>(answeredQuestionsAccumulator));
122     testPlayer = answerQuestionService.answerQuestion(playerId:"1", answeredQuestionDTO);
123
124     // Add the current answered question to the accumulator
125     answeredQuestionsAccumulator
126         .add(testPlayer.getAnsweredQuestions().get(testPlayer.getAnsweredQuestions().size() - 1));
127
128     double actualAverageDeviation = testPlayer.getAverageDeviation();
129
130     // Assert that the calculated average deviation matches the expected average deviation within a
131     // tolerance of 0.01 (two decimal places)
132     assertEquals(expectedAverageDeviation, actualAverageDeviation, delta:0.01);
133 }
134
135 }
136
137

```

ScoreCalculationTest

Dieser Modultest stellt sicher, dass die Punktberechnung korrekt funktioniert.

setUpForScoreTest(): Diese Methode wird vor jedem Test ausgeführt, um die Testdaten vorzubereiten und das Verhalten der Mock-Repositories zu konfigurieren.

answerQuestionTotalScoreTest(): Dieser Test überprüft die Gesamtpunktberechnung basierend auf den Antworten des Spielers.

answerQuestionIndividualScoreTest(): Dieser Test überprüft die individuelle Punkteberechnung für jede beantwortete Frage.

```
26 @ExtendWith(MockitoExtension.class)
27 class ScoreCalculationTest {
28
29     // Inject the mocks into the AnswerQuestionService instance for testing
30     @InjectMocks
31     private AnswerQuestionService answerQuestionService;
32
33     // Mock the PlayerRepository and QuestionRepository dependencies
34     @Mock
35     private PlayerRepository playerRepository;
36
37     @Mock
38     private QuestionRepository questionRepository;
39
40     private Player testPlayer;
41     private Question testQuestion;
42
43     @BeforeEach
44     public void setUpForScoreTest() {
45         testPlayer = new Player(email:"test@example.com", username:"TestUser");
46         testPlayer.setId(id:"1");
47         testPlayer.setAnsweredQuestions(new ArrayList<>());
48
49         testQuestion = new Question(id:"1", questionText:"Test question", correctAnswer:42, level:"LEVEL_1");
50
51         // Configure the mock repositories to return the test data
52         when(playerRepository.findById("1")).thenReturn(Optional.of(testPlayer));
53         when(questionRepository.findById("1")).thenReturn(Optional.of(testQuestion));
54
55         when(playerRepository.save(any(type:Player.class))).thenAnswer(i -> i.getArguments()[0]);
56     }
57
58     private static Stream<Arguments> provideScoreTestData() {
59         return Stream.of(
60             // playerAnswer, expectedScore
61             Arguments.of(42, 100),
62             Arguments.of(43, 99),
63             Arguments.of(47, 89),
64             Arguments.of(53, 75));
65     }
66
67     @ParameterizedTest
68     @MethodSource("provideScoreTestData")
69     void answerQuestionTotalScoreTest(int playerAnswer, int expectedScore) {
70         AnsweredQuestionDTO answeredQuestionDTO = new AnsweredQuestionDTO(questionId:"1", playerAnswer);
71
72         testPlayer = answerQuestionService.answerQuestion(playerId:"1", answeredQuestionDTO);
73
74         int actualTotalScore = testPlayer.getScore();
75
76         // Assert that the calculated total score matches the expected score
77         assertEquals(expectedScore, actualTotalScore);
78     }
79
80     @ParameterizedTest
81     @MethodSource("provideScoreTestData")
82     void answerQuestionIndividualScoreTest(int playerAnswer, int expectedScore) {
83         AnsweredQuestionDTO answeredQuestionDTO = new AnsweredQuestionDTO(questionId:"1", playerAnswer);
84
85         testPlayer = answerQuestionService.answerQuestion(playerId:"1", answeredQuestionDTO);
86
87         // Get the score of the last answered question.
88         int actualQuestionScore = testPlayer.getAnsweredQuestions()
89             .get(testPlayer.getAnsweredQuestions().size() - 1).getQuestionScore();
90
91         // Assert that the calculated question score matches the expected score
92         assertEquals(expectedScore, actualQuestionScore);
93     }
94 }
```


INTEGRATIONTESTS

In der EndpointTests Klasse werden eine Reihe von Integrationstests ausgeführt. Diese stellen sicher, dass die REST-API wie erwartet funktioniert. Es werden Tests für die Erstellung eines Spielers (testCreatePlayer), das Abrufen von Spielerinformationen (testGetPlayer und testGetPlayerById), das Löschen eines Spielers (testDeletePlayer) und das Abrufen von Fragen (testGetQuestions) durchgeführt.

Diese Tests verwenden das MockMvc Framework, um HTTP-Anfragen an die API zu senden und die Antworten zu validieren. Wir verwenden auch die @WithMockUser-Annotation, um sicherzustellen, dass die Zugriffssteuerung unserer API korrekt funktioniert.

Zusätzlich implementiert TestSecurityConfig einen Mock JwtDecoder für die Authentifizierung in unseren Tests.

EndpointTests

testCreatePlayer(): Dieser Test überprüft, ob ein neuer Spieler korrekt erstellt und in der Datenbank gespeichert werden kann.

testGetPlayer(): Dieser Test überprüft, ob die Daten eines Spielers korrekt abgerufen werden können.

testGetPlayerById(): Dieser Test überprüft, ob die Daten eines bestimmten Spielers über seine ID korrekt abgerufen werden können.

testDeletePlayer(): Dieser Test überprüft, ob ein Spieler korrekt aus der Datenbank gelöscht werden kann.

testGetQuestions(): Dieser Test überprüft, ob die Fragen korrekt abgerufen werden können.

```

33 @SpringBootTest
34 @Import(TestSecurityConfig.class)
35 @AutoConfigureMockMvc
36 @TestMethodOrder(OrderAnnotation.class)
37 class EndpointTests {
38
39     @Autowired
40     private MockMvc mockMvc;
41
42     @Autowired
43     private ObjectMapper objectMapper;
44
45     @Autowired
46     private PlayerRepository playerRepository;
47
48     private static final String testName = "testPlayerName";
49     private static final String testEmail = "testEmail@example.com";
50
51     @Test
52     @Order(1)
53     @WithMockUser
54     void testCreatePlayer() throws Exception {
55         // create a test player and convert to json
56         Player player = new Player(testEmail, testName);
57         var jsonBody = objectMapper.writeValueAsString(player);
58
59         // POST json to service with authorization header
60         mockMvc.perform(post(urlTemplate: "/api/player")
61             .contentType(MediaType.APPLICATION_JSON)
62             .content(jsonBody)
63             .header(HttpHeaders.AUTHORIZATION, ...values: "Bearer token"))
64             .andDo(print())
65             .andExpect(status().isCreated())
66             .andReturn();
67
68         // Save the created player into class level variable
69
70     }
71
72     @Test
73     @Order(2)
74     @WithMockUser
75     void testGetPlayer() throws Exception {
76         mockMvc.perform(get(urlTemplate: "/api/player")
77             .header(HttpHeaders.AUTHORIZATION, ...values: "Bearer token"))
78             .andDo(print())
79             .andExpect(status().isOk())
80             .andReturn();
81     }
82
83     @Test
84     // @Disabled
85     @Order(3)

```

```

86     @WithMockUser
87     void testGetPlayerById() throws Exception {
88         Player player = playerRepository.findFirstByEmail(testEmail);
89         System.out.println("////////////////////" + player.toString());
90         // Use the class level variable 'player'
91         mockMvc.perform(get("/api/player/" + player.getId())
92             .contentType(MediaType.APPLICATION_JSON)
93             .header(HttpHeaders.AUTHORIZATION, ..values:"Bearer token"))
94             .andDo(print())
95             .andExpect(status().isOk())
96             .andReturn();
97     }
98
99     @Test
100     // @Disabled
101     @Order(4)
102     @WithMockUser
103     void testDeletePlayer() throws Exception {
104         Player player = playerRepository.findFirstByEmail(testEmail);
105         // Use the class level variable 'player'
106         mockMvc.perform(delete("/api/player/" + player.getId())
107             .contentType(MediaType.APPLICATION_JSON)
108             .header(HttpHeaders.AUTHORIZATION, ..values:"Bearer token"))
109             .andDo(print())
110             .andExpect(status().isOk())
111             .andReturn();
112         // Ensure that the Player was deleted
113         Player result = playerRepository.findFirstByEmail(testEmail);
114         assertNull(result);
115     }
116
117     @Test
118     @Order(5)
119     @WithMockUser
120     void testGetQuestions() throws Exception {
121         mockMvc.perform(get(urlTemplate:"/api/question")
122             .header(HttpHeaders.AUTHORIZATION, ..values:"Bearer token"))
123             .andDo(print())
124             .andExpect(status().isOk())
125             .andExpect(content().contentTypeCompatibleWith(MediaType.APPLICATION_JSON))
126             .andReturn();
127     }
128 }
129

```

TestSecurityConfig

Diese Konfigurationsklasse stellt sicher, dass während der Tests gültige JSON Web Tokens (JWTs) erstellt werden. Dies ist wichtig, da die Anwendung Authentifizierung und Autorisierung durch JWTs implementiert. Dies stellt sicher, dass während der Tests die Authentifizierungs- und Autorisierungslogik korrekt simuliert wird.

```

13 @TestConfiguration
14 public class TestSecurityConfig {
15
16     @Bean
17     public JwtDecoder jwtDecoder() {
18         return new JwtDecoder() {
19             @Override
20             public Jwt decode(String token) {
21                 return jwt();
22             }
23         };
24     }
25
26     public Jwt jwt() {
27         Map<String, Object> claims = new HashMap<>();
28         claims.put(key:"sub", value:"test");
29         claims.put(key:"user_roles", Arrays.asList(...:"admin"));
30         return new Jwt(
31             tokenValue:"AUTH0_TOKEN",
32             Instant.now(),
33             Instant.now().plusSeconds(secondsToAdd:30),
34             Map.of(k1:"alg", v1:"none"),
35             claims
36         );
37     }
38 }
39

```

USER-TESTS

Um die Usability der App zu Testen haben vier Personen eine der folgenden Testszenarien erhalten:

Testszenario 1:

<p style="text-align: center;"><Ausgangslage></p> <p>Sie sind ein Spieler der Applikation. Sie wollen das Spiel durchspielen.</p>
<p>Aufgabe 1: Sie möchten einen Account erstellen.</p>
<p>Aufgabe 2: Sie möchten Level 1 abschliessen.</p>
<p>Aufgabe 3: Sie möchten Level 2 spielen, während dem Spiel möchten Sie zurück auf die Homepage und danach weiterspielen.</p>
<p>Aufgabe 4: Sie möchten nun einen Rang im Leaderboard von Completed Players erhalten.</p>

Testszenario 2:

Aufgabe 1:

Sie möchten eine Liste aller Spieler sehen.

Aufgabe 2:

Sie möchten einen Spieler hinzufügen.

Aufgabe 3:

Sie möchten einen Spieler löschen.




Für jede Testperson wird währenddem sie das Testszenario durchmacht Beobachtungen auf dem Feedback Grid ausgefüllt. Im Anschluss wird zusammen mit der Testperson über den Gesamteindruck diskutiert und ebenfalls im Feedback Grid protokolliert.

Testszenario 1:

Person 1:

Was hat gut funktioniert? Was hat gefallen?	Was hat nicht/schlecht funktioniert? Was hat gestört? Was hat gefehlt (Funktionen, Optionen, Infos ...)?
<ul style="list-style-type: none">- Design- Interessantes Konzept- Leaderboard 	<ul style="list-style-type: none">- Levels nicht anklickbar- Seite gecrasht bei Level 4- Masseinheiten anderst anzeigen 
Welche neuen Ideen, Anforderungen sind aufgekommen?	? Was war unklar (Abfolge, Benennungen, Worte, Texte ...)? Welche Fragen sind aufgetaucht?
<ul style="list-style-type: none">- Levels klickbar machen 	<ul style="list-style-type: none">- Fragestellungen lassen teils Interpretationsraum 

Person 2:



<p><i>Was hat gut funktioniert? Was hat gefallen?</i></p> <ul style="list-style-type: none">• ...- Design- Timer bei Fragen- Einfaches Passwort 	<p><i>Was hat nicht/schlecht funktioniert? Was hat gestört? Was hat gefehlt (Funktionen, Optionen, Infos ...)?</i></p> <ul style="list-style-type: none">• ...- Manchmal Seite refreshen (bei Leaderboard)- Gewisse Backgroundinfos zu gewissen Fragen 
<p><i>Welche neuen Ideen, Anforderungen sind aufgekommen?</i></p>  <ul style="list-style-type: none">• ...- Username ändern	<p><i>? Was war unklar (Abfolge, Benennungen, Worte, Texte ..)? Welche Fragen sind aufgetaucht?</i></p> <ul style="list-style-type: none">• ...

Testszenario 2:

Person 3

<p><i>Was hat gut funktioniert? Was hat gefallen?</i></p> <ul style="list-style-type: none">• ...- Pagination bei Players- Automatischer refresh nach löschen 	<p><i>Was hat nicht/schlecht funktioniert? Was hat gestört? Was hat gefehlt (Funktionen, Optionen, Infos ...)?</i></p> <ul style="list-style-type: none">• ... 
<p><i>Welche neuen Ideen, Anforderungen sind aufgekommen?</i></p>  <ul style="list-style-type: none">• ...	<p><i>? Was war unklar (Abfolge, Benennungen, Worte, Texte ..)? Welche Fragen sind aufgetaucht?</i></p> <ul style="list-style-type: none">• ...- Alle Spieler ansehen, ging auf Leaderboard zuerst

Person 4

Was hat gut funktioniert? Was hat gefallen?	Was hat nicht/schlecht funktioniert? Was hat gestört? Was hat gefehlt (Funktionen, Optionen, Infos ...)?
<ul style="list-style-type: none">...- Design 	<ul style="list-style-type: none">...- Input Felder sind nicht geleert nachdem Spieler geaddet wird 
Welche neuen Ideen, Anforderungen sind aufgekommen?	? Was war unklar (Abfolge, Benennungen, Worte, Texte ..)? Welche Fragen sind aufgetaucht?
<ul style="list-style-type: none">...- Inputfelder leeren nach Add Player 	<ul style="list-style-type: none">... 

RESULTATE DER USER-TESTS, DISKUSSION DER ERGEBNISSE

In der Phase der Nutzertests konnten eine Reihe von Rückmeldungen und Anmerkungen zu unserer Guesst Anwendung gesammelt werden. Im Folgenden werden die wichtigsten Erkenntnisse aus den Tests und deren Auswirkungen auf der Anwendung diskutiert.

Testergebnisse:

Levels nicht anklickbar

Ein Feedback betraf die der Levelauswahl. Der Benutzer wollte das auf der Startseite die Levels angeklickt werden können und evtl. abgeschlossene Levels anzuschauen was für Schätzungen abgegeben wurden.

Masseinheiten anderst anzeigen

Eine weitere Anmerkung war die Darstellung der Masseinheiten. Es wurde festgestellt, dass die Darstellung und die Art und Weise, wie Masseinheiten in der Anwendung dargestellt wurden, für einige Benutzer verwirrend war.

Fragestellungen lassen teils Interpretationsraum

Es wurde auch Feedback zu den Fragen in der Anwendung gegeben. Ein Benutzer fand die Fragestellungen teils mehrdeutig und offen für Interpretationen, was dazu führte, dass sie Schwierigkeiten hatten, die Fragen zu verstehen und zu beantworten.

Manchmal Seite refreshen (bei Leaderboard)

Das Leaderboard aktualisiert manchmal nicht, es sei denn, die Benutzer aktualisierten die ganze Seite.

Gewisse Backgroundinfos zu gewissen Fragen

Ein Benutzer äusserte den Wunsch nach mehr Hintergrundinformationen zu bestimmten Fragen. Diese Information könnte ihnen dabei helfen, die Frage besser zu verstehen und eine informierte Schätzung abzugeben.

Diskussion der Ergebnisse:

Die User-Tests der GuessIt-Anwendung haben wichtige Bereiche zur Verbesserung aufgezeigt. Hier sind eventuelle Lösungen für die identifizierten Probleme:

Levels nicht anklickbar

Die bereits abgeschlossenen Levels könnten anklickbar sein, um seine alten Schätzungen ersichtlich zu machen.

Masseinheiten anderst anzeigen

Eine einheitliche und klare Darstellung der Masseinheiten könnte eingeführt werden, möglicherweise mit kurzen Erläuterungen zu jeder Einheit oder das Eingabefeld nur auf die gewollte Masseinheit abstimmen.

Fragestellungen lassen teils Interpretationsraum

Eine gründlichere Überprüfung und Validierung der Fragen könnte sicherstellen, dass sie eindeutig und leicht verständlich sind.

Manchmal Seite refreshen (bei Leaderboard)

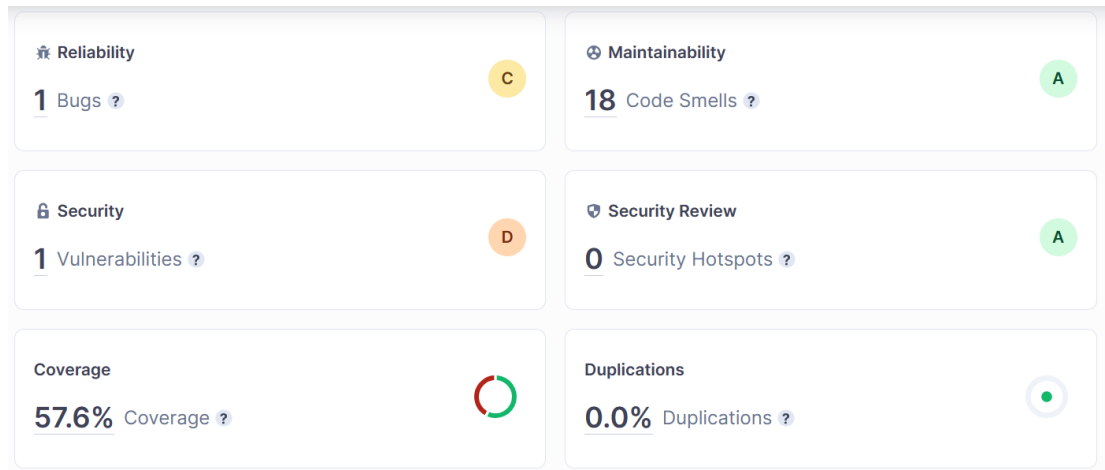
Ein automatisches Aktualisierungsfeature könnte eingeführt werden, um das manuelle Aktualisieren der Leaderboard-Seite zu vermeiden.

Gewisse Backgroundinfos zu gewissen Fragen

Kurze Informationsblöcke könnten bereitgestellt werden, um zusätzlichen Kontext oder Informationen zu den Fragen zu liefern.

Codeanalyse mit SonarCube

Der Code wurde schlussendlich noch mit SonarCube analysiert und ergab folgendes:



BUGS

Es gibt einen Bug:

```

return new JwtAuthenticationToken(jwt, Stream.concat(
  JwtGrantedAuthoritiesConverter().convert(jwt)
  .stream(), extractResourceRoles(jwt).stream())
  .collect(Collectors.toSet()));

```

A "NullPointerException" could be thrown; "convert()" can return null.

Um dies zu beheben, müsste man vor dem Aufruf von stream() prüfen, ob convert(jwt) null zurückgibt. Wenn dass der Fall wäre, könnte man dann dort gleich damit umgehen. Dies wird allerdings erst zu einem späteren Zeitpunkt behoben.

CODE SMELLS

Im MongoDB Controller war das Exception Handling leer:

```

28 // verify retrived document
29 if (read != null && read.get("time").toString().equals(time
   .toString())) {
30     return new ResponseEntity<>("Connection ok", HttpStatus.OK);
31 }
32 } catch (Exception e) {
33 }
34 return new ResponseEntity<>("Connection failed", HttpStatus
   .INTERNAL_SERVER_ERROR);
35 }

```

Either remove or fill this block of code.

Dies wurde behoben, indem eine Error Nachricht durch den SLF4J Logger erscheint.

```
} catch (Exception e) {  
    LOGGER.error(msg:"Error while testing MongoDB", e);  
}
```

Des weiteren waren viele Code Smells dieselben:

Remove this 'public' modifier.	No tags
<input checked="" type="radio"/> Code Smell <input type="radio"/> Open <input type="radio"/> Info Svenson-Maximus	2min effort • 27 days ago
Remove this 'public' modifier.	No tags
<input checked="" type="radio"/> Code Smell <input type="radio"/> Open <input type="radio"/> Info Svenson-Maximus	2min effort • 26 days ago
Remove this 'public' modifier.	No tags
<input checked="" type="radio"/> Code Smell <input type="radio"/> Open <input type="radio"/> Info Svenson-Maximus	2min effort • 11 days ago
Remove this 'public' modifier.	No tags
<input checked="" type="radio"/> Code Smell <input type="radio"/> Open <input type="radio"/> Info Svenson-Maximus	2min effort • 8 days ago

src/test/java/ch/zhaw/guess/EndpointTests.java

Remove this 'public' modifier.	No tags
<input checked="" type="radio"/> Code Smell <input type="radio"/> Open <input type="radio"/> Info Not assigned	2min effort • 1 hour ago

Diese wurden alle behoben nach erneuter Analyse sieht es nun so aus:



Fazit

STAND DER IMPLEMENTATION

Der derzeitige Prototyp zeigt eine starke Leistung, mit allen geplanten Funktionen, die erfolgreich implementiert und funktionsfähig sind. Auf Basis der durchgeführten Usertests konnten wertvolle Erkenntnisse gewonnen werden, die potenzielle Erweiterungen und Optimierungen für das Web-App aufzeigen:

Anwender schlagen teilweise äusserst interessante Schätzfragen vor, die als potenzielle Inhalte für künftige Levels berücksichtigt werden könnten. Als Admin könnte man z.B. dann diese Fragen einsehen und gute für weitere Levels integrieren. Dabei wäre es jedoch notwendig, das Scoring-System entsprechend zu adaptieren, um die faire Bewertung der Spielerleistung zu gewährleisten.

Die Integration weiterer Levels könnte das Spielerlebnis für bereits fertige Spieler verlängern und sie zur Rückkehr bewegen.

Eine Anpassung der Schwierigkeit nach Level könnte ebenfalls in Betracht gezogen werden. Aktuell sind die Schätzfragen zufällig auf die Levels verteilt und nicht nach Schwierigkeitsgrad geordnet. Eine ansteigende Schwierigkeitskurve, die mit dem Fortschreiten der Levels steigt, könnte das Spiel noch anspruchsvoller und herausfordernder gestalten.

Letztendlich sollten die Änderungen gemäss den User-Tests noch integriert werden.

PERSÖNLICHES FAZIT

Die Entwicklung dieses Prototyps war eine aufschlussreiche Erfahrung, die es mir ermöglichte, die Praktikabilität meiner Kenntnisse und Fähigkeiten in einem realen Projektumfeld zu testen. Ich bin zufrieden mit dem erreichten Stand der Implementierung und freue mich darauf, die erkannten Verbesserungsmöglichkeiten zu nutzen und eventuell die Web-App weiterzuentwickeln.