

# Dokumentation

# Finish This!

*Finish This!*

Individuelle Abschlussarbeit BLJ

Levi Agostinho Horta

Adnovum AG

Unterschrift: *Levi H.*

## Inhaltsverzeichnis

1.	Änderungstabelle.....	3
2.	Einleitung .....	3
2.1	Aufgabenstellung und Projektbeschreibung .....	4
2.2	Benutzte Technologie .....	4
2.3	Bekannte Risiken.....	5
2.4	Projektziele.....	5
3.	Planung .....	6
3.1	Terminplan .....	6
3.2	Risikoanalyse.....	7
	3.3 Entscheidungsmatrix.....	7
4.	Hauptteil .....	8
4.1	Umsetzung und Zwischenschritte .....	8
4.1.1	Phase 1: Grundlagen (Tag 1-3) .....	8
4.1.2	Phase 2: Multiplayer-Features (Tag 4-8).....	8
4.1.3	Phase 3: Finalisierung (Tag 9-12).....	9
4.2	Ergebnis der Arbeit .....	10
4.2.1	Kernfunktionalitäten .....	11
4.2.2	Technische Leistungen .....	12
4.3	Funktionsbeschreibung .....	12
4.3.1	Komponenten.....	12
4.3.2	Spielablauf.....	12
4.4	Arbeitsjournal .....	13
4.4.1	Tag 1: 02.06.2025 .....	13
4.4.2	Tag 2: 03.06.2025 .....	13
4.4.3	Tag 3: 04.06.2025 .....	13
4.4.4	Tag 4: 10.06.2025 .....	14
4.4.5	Tag 5: 11.06.2025 .....	14
4.4.6	Tag 6: 16.06.2025 .....	14
4.4.7	Tag 7: 17.06.2025 .....	15
4.4.8	Tag 8: 18.06.2025 .....	15
4.4.9	Tag 9: 20.06.2025 .....	15

4.4.10	Tag 10: 23.06.2025 .....	16
4.4.11	Tag 11: 24.06.2025 .....	16
4.4.12	Tag 12: 25.06.2025 .....	16
4.5	Testplan.....	17
4.6	Herausforderungen und Planänderungen.....	18
4.7	Flussdiagramm .....	19
5.	Anhang.....	19
5.1	Glossar.....	19
5.2	Codeausschnitte.....	21
5.2.1	Emoji-Voting System .....	21
5.2.2	Polling-Mechanismus für Echtzeit-Updates .....	22
5.2.3	Datenbankabfrage für Sätze mit Votes .....	22
5.3	Screenshot-App .....	23
5.4	KI-Chat-Auszüge .....	24
5.5	Quellen und Literatur.....	24
5.6	Bildverzeichnis.....	24
5.7	Checklist .....	25
5.8	Github sowie Spiel-Link .....	25
6.	Fazit und Reflexion .....	26

## 1. Änderungstabelle

<b>Wer</b>	<b>Was</b>	<b>Wann</b>	<b>Version</b>
Levi Agostinho Horta	Dokumentation starten	02.06.2025	1.0
Levi Agostinho Horta	Einleitung und Planung geschrieben	03.06.2025	1.1
Levi Agostinho Horta	Entscheidungsmatrix und Doku erweitert mit Bildern.	04.06.2025	1.2
Levi Agostinho Horta	Arbeitsjournal erweitert	10.06.2025	1.3
Levi Agostinho Horta	Arbeitsjournal erweitert	11.06.2025	1.4
Levi Agostinho Horta	Arbeitsjournal erweitert	16.06.2025	1.5
Levi Agostinho Horta	Arbeitsjournal erweitert	17.06.2025	1.6
Levi Agostinho Horta	Arbeitsjournal erweitert	18.06.2025	1.7
Levi Agostinho Horta	Arbeitsjournal erweitert	20.06.2025	1.8
Levi Agostinho Horta	Arbeitsjournal und Titelblatt erweitert	23.06.2025	1.9
Levi Agostinho Horta	Arbeitsjournal erweitert	24.06.2025	2.0
Levi Agostinho Horta	Vollständige Dokumentation mit allen Anforderungen	25.06.2025	2.1

## 2. Einleitung

### Einleitung

Im Rahmen meiner individuellen Abschlussarbeit BLJ habe ich eine kreative Web-App mit dem Namen „Finish This!“ entwickelt. Das Ziel bestand darin, eine Multiplayer-Anwendung zu schaffen, bei der mehrere Spieler gleichzeitig an einem Spiel teilnehmen können. Zu Beginn einer Runde erhält jeder Spieler einen zufälligen Satzanfang, den er oder sie dann kreativ zu Ende schreiben muss. Die fertigen Sätze werden abschliessend von den anderen Spielern mithilfe von Emojis bewertet, beispielsweise 😊 für besonders kreative Texte oder 🥰 für eher lustige oder Beiträge, die keinen Sinn ergeben.

Dadurch entsteht ein interaktives Spiel, bei dem Kreativität, Humor und Teamgeist gefragt sind. Gleichzeitig habe ich damit ein Projekt umgesetzt, das wichtige technische Aspekte der Webentwicklung abdeckt: ein Frontend mit React und TypeScript, ein eigenes Backend mit Node.js und Express sowie eine PostgreSQL-Datenbank zur Speicherung von Spielern, Räumen und Votes.

## 2.1 Aufgabenstellung und Projektbeschreibung

### Aufgabenstellung und Projektbeschreibung

Im Rahmen meiner individuellen Abschlussarbeit BLJ hatte ich die Aufgabe, eine eigene Webanwendung zu entwickeln und dabei ein vollständiges Softwareprojekt durchzuführen – von der Idee über die Umsetzung bis hin zur Dokumentation. Mein Ziel war es, ein kreatives und interaktives Multiplayer-Spiel zu programmieren, das sowohl technisch als auch designtechnisch überzeugt.

Die Projektidee war eine Web-App namens „Finish This!“, bei den mehrere Spielern gleichzeitig spielen können. Zu Beginn bekommt jeder Spieler einen zufälligen Satzanfang angezeigt. Diesen müssen sie kreativ mit einer eigenen Endung vervollständigen. Anschliessend bewerten alle Spieler gegenseitig ihre Texte mit Emojis. Dadurch entsteht ein spielerisches Erlebnis, das Spass macht und bei dem man Punkte sammeln kann.

Die App sollte die folgenden Bereiche abdecken: ein responsives Frontend mit React und TypeScript, ein eigenes Backend mit Node.js und Express sowie eine PostgreSQL-Datenbank zur Speicherung der Spiel- und Bewertungsdaten. Darüber hinaus sollte die App auf der Cloud-Plattform Render bereitgestellt werden. Zusätzlich war es meine Aufgabe, alle Anforderungen an Benutzerfreundlichkeit, Design, Sicherheit und Performance zu berücksichtigen. Die Dokumentation sollte so geschrieben sein, dass auch jemand, der nicht am Projekt beteiligt war, alles nachvollziehen kann.

## 2.2 Benutzte Technologie

### Benutze Technologie

Die Umsetzung erfolgte mit folgenden Webtechnologien:

- **Frontend:** React mit TypeScript für typsichere Entwicklung
- **Backend:** Node.js mit Express.js für die API-Entwicklung
- **Datenbank:** PostgreSQL für zuverlässige Datenspeicherung
- **Deployment:** Render für Cloud-Hosting
- **Versionskontrolle:** Git/GitHub für Projektverwaltung

## 2.3 Bekannte Risiken

### Bekannte Risiken

Schon am Anfang vom Projekt habe ich mir überlegt, was alles schwierig werden könnte und wo es Probleme geben könnte. Ein mögliches Risiko war die Web-Socket-Kommunikation, weil ich eigentlich wollte, dass das Spiel komplett in Echtzeit läuft. Aber da ich Web Sockets noch nicht so gut kenne und es schnell kompliziert werden kann mit mehreren Clients gleichzeitig, habe ich das Risiko erkannt und stattdessen auf eine einfachere Lösung mit Polling gesetzt.

Ein weiteres Risiko war das Zeitmanagement. Ich wusste, dass ich nicht unendlich Zeit habe und mir gut überlegen muss, welche Funktionen wirklich wichtig sind. Es hätte passieren können, dass ich zu viele Ideen habe und nicht alles rechtzeitig fertigbekomme. Deshalb habe ich früh angefangen, eine klare Planung zu machen und mich auf die Hauptfunktionen konzentriert.

Auch das UI/UX-Design war ein Thema, weil es gar nicht so einfach ist, ein Spiel zu bauen, das für alle verständlich ist. Ich hatte die Befürchtung, dass es am Anfang zu kompliziert wirkt oder unübersichtlich ist. Darum habe ich versucht, das Design möglichst simpel und benutzerfreundlich zu machen – z. B. mit klaren Buttons, logischen Farben und wenig Ablenkung.

Bei der Datenbankintegration und Performance war mir von Anfang an klar, dass ich schauen muss, wie ich die Abfragen schreibe, damit es später nicht langsam wird. Vor allem, wenn viele Spieler gleichzeitig aktiv ist, kann das die Performance beeinflussen. Ich habe deshalb gelernt, wie man mit PostgreSQL optimiert arbeiten kann.

Ein weiteres Risiko war die Deployment-Konfiguration. Ich wusste nicht, wie schwer es sein wird, die App auf Render zu deployen.

## 2.4 Projektziele

### Projektziele

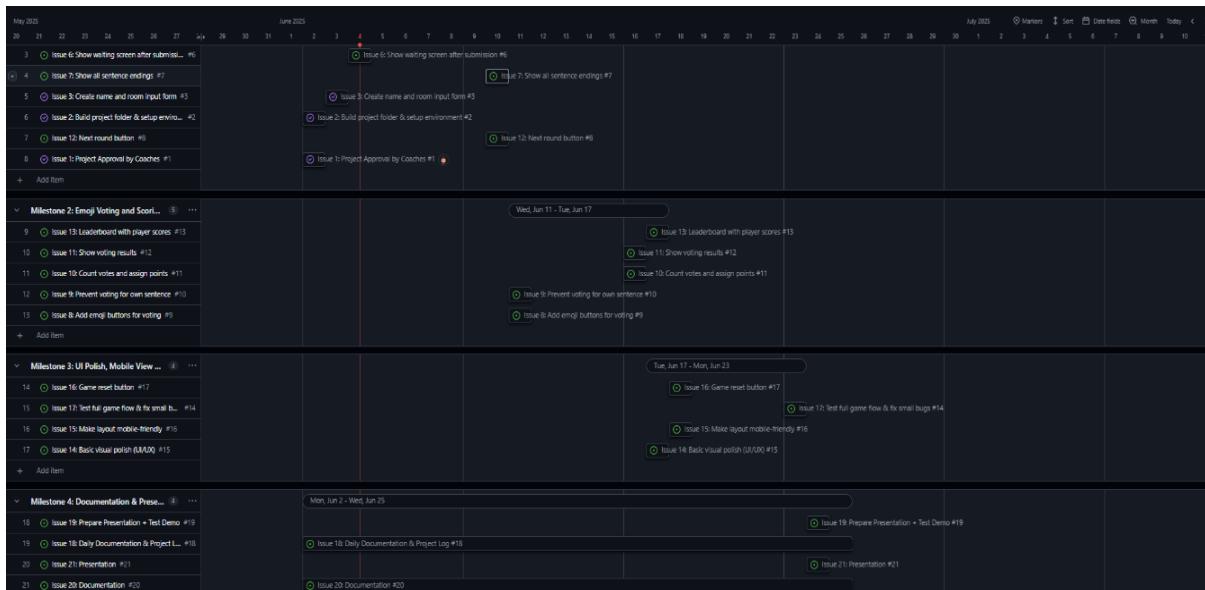
- Entwicklung einer funktionsfähigen Multiplayer-Web-Anwendung
- Implementierung eines benutzerfreundlichen Voting-Systems
- Erstellung einer responsiven Benutzeroberfläche
- Integration einer Datenbank für persistente Datenspeicherung
- Deployment der Anwendung auf einer Cloud-Plattform

## 3. Planung

### 3.1 Terminplan

#### Terminplan

Tag	Umsetzung	Status
Tag 1	Projektsetup, Grundstruktur, GitHub eingerichtet	✓
Tag 2	Join-Form mit State	✓
Tag 3	Satzanfang und Eingabe Feld	✓
Tag 4	Next Round Button und Emojis Voting	✓
Tag 5	Verbindung von Frontend und Backend	✓
Tag 6	Ranking von Sätzen und Podium von Spielern	✓
Tag 7	UI und Design angepasst sowie benutzerfreundlicher gemacht	✓
Tag 8	PostgreSQL implementiert und Voting und Ranking System funktionsfähig gemacht	✓
Tag 9	Warte Zimmer implementiert und Benutzerfreundlichkeit verbessert	✓
Tag 10	Deployment auf Render und Name von beigetretenem Spieler sichtbar machen.	✓
Tag 11	Präsentation anfangen und fertig machen sowie Doku	✓
Tag 12	Bug fixes und Präsentationsvorbereitung	✓



1. Abbildung von Roadmap

## 3.2 Risikoanalyse

### Risikoanalyse

Risiko	Wahrscheinlichkeit	Auswirkung	Massnahmen
Zeitüberschreitung	Mittel	Hoch	Priorisierung der Kernfeatures
Backend-Komplexität	Hoch	Mittel	Schrittweise Implementierung
Deployment-Probleme	Mittel	Hoch	Frühe Tests der Deployment-Pipeline
UI/UX-Herausforderungen	Niedrig	Mittel	Styling-Verbesserungen

```

Deploy failed for 67a65e4: Merge pull request #44 from Hortalevi/Tag-10 db.js cleaning
  ✘ Root directory 'backend' does not exist, please check settings
  June 23, 2025 at 10:11 AM
    ↗ Rollback

Deploy started for 67a65e4: Merge pull request #44 from Hortalevi/Tag-10 db.js cleaning
  ↗ Start command updated
  June 23, 2025 at 10:11 AM

```

2. Deployment Test vom Backend in Render

## 3.3 Entscheidungsmatrix

### Entscheidungsmatrix

Option	Aufwand	Nutzen	Komplexität	Technische Machbarkeit	Entscheid
Satzspiel mit Voting	Mittel	Hoch	Mittel	Hoch	Gewählt
WebSocket-Spiel	Hoch	Hoch	Hoch	Mittel	Später
Lokale Spieleingabe	Niedrig	Mittel	Niedrig	Sehr hoch	Umgesetzt
React + TypeScript	Mittel	Hoch	Mittel	Hoch	Gewählt
PostgreSQL Datenbank	Mittel	Hoch	Mittel	Hoch	Gewählt
Mobile-First Design	Hoch	Hoch	Hoch	Mittel	Implementiert

## 4. Hauptteil

### 4.1 Umsetzung und Zwischenschritte

#### Umsetzung und Zwischenschritte

Die App wurde mit React (TypeScript) umgesetzt. Zunächst wurde das Projekt strukturiert und das Frontend gestartet. Anschliessend folgten Join-Form, die Speicherung von Eingaben und die Anzeige eines zufälligen Satzanfangs.

#### 4.1.1 Phase 1: Grundlagen (Tag 1-3)

##### Phase 1: Grundlagen (Tag 1-3)

In Phase 1 (Tag 1 bis 3) habe ich das Projekt mit React und TypeScript gestartet. Das hilft mir, sauber mit Komponenten zu arbeiten und direkt Typen zu verwenden, wodurch Fehler vermieden werden. Ich habe zwei Hauptkomponenten erstellt: das Join-Formular, in das man seinen Nickname eingibt und einem Raum beitritt, sowie den Game-Screen, in dem das Spiel dann läuft. Damit alles flüssig funktioniert, habe ich den Zustand mit React-Hooks lokal verwaltet, beispielsweise für den aktuellen Satz, die Runde oder die Spieler, die bereits etwas abgeschickt haben. Ausserdem habe ich eine Funktion eingebaut, die zufällig einen Satzanfang auswählt, den die Spieler dann weiterschreiben können.

#### 4.1.2 Phase 2: Multiplayer-Features (Tag 4-8)

##### Phase 2: Multiplayer-Features (Tag 4-8)

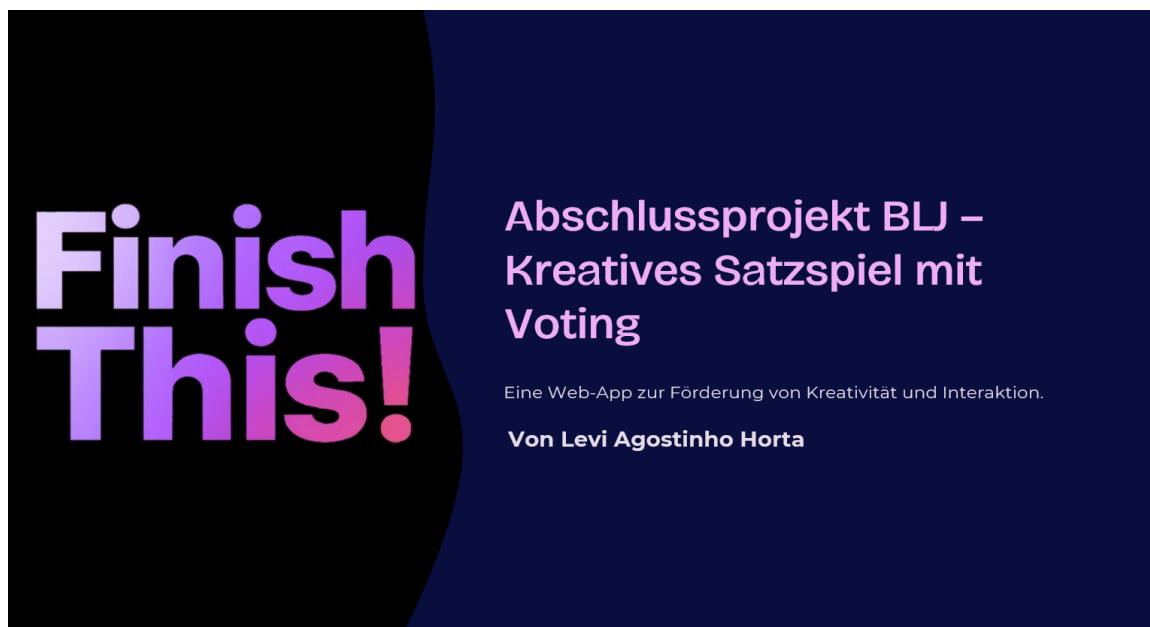
In Phase 2 des Projekts (Tag 4 bis 8) habe ich die Multiplayer-Funktionen integriert. Dazu habe ich das Backend mit Node.js und Express verbunden, damit die Spieldaten richtig verarbeitet werden. Ausserdem habe ich ein Emoji-Voting-System programmiert, mit dem man die Sätze anderer mit Emojis wie 😊, 🤪, 🤓 oder 💩 bewerten kann. Je nach gewählter Emoji gibt es unterschiedlich viele Punkte. Ausserdem habe ich eine PostgreSQL-Datenbank eingebunden, um Räume, Spieler, Sätze und Bewertungen zu speichern. Zudem habe ich ein Ranking- und Punktesystem umgesetzt, das anzeigt, wer die meisten Punkte hat. Für alle Spielfunktionen habe ich passende API-Endpunkte programmiert, damit das Frontend mit dem Server kommunizieren kann.

#### 4.1.3 Phase 3: Finalisierung (Tag 9-12)

##### Phase 3: Finalisierung (Tag 9-12)

In Phase 3 (Tag 9 bis 12) habe ich das Projekt fertiggestellt und noch einmal verbessert. Ich habe ein Wartezimmer eingebaut, in dem man sieht, wer sich bereits im Raum befindet, und in dem der Countdown startet, sobald genügend Spieler anwesend sind. Ausserdem habe ich Reset-Funktionen programmiert, mit denen man eine neue Runde starten oder das Spiel sauber beenden kann. Beim Deployment habe ich darauf geachtet, dass alles gut auf Render läuft und schnell lädt. Ausserdem habe ich das Design verbessert, d. h., ich habe die Benutzeroberfläche übersichtlicher gestaltet und kleine Details angepasst. Schliesslich habe ich dafür gesorgt, dass alles auch auf dem Handy gut aussieht und funktioniert.

Am letzten Tag habe ich schliesslich die Dokumentation fertiggestellt, in der ich alle wichtigen Aspekte des Projekts erläutert habe – vom Aufbau bis zu den Funktionen. Ausserdem habe ich eine Präsentation vorbereitet, um das Projekt vorzustellen und zu zeigen, wie es funktioniert. Dafür habe ich Screenshots, Codeausschnitte und eine kurze Live-Demo eingebaut, die veranschaulichen, was ich in den zwei Wochen alles gemacht habe.



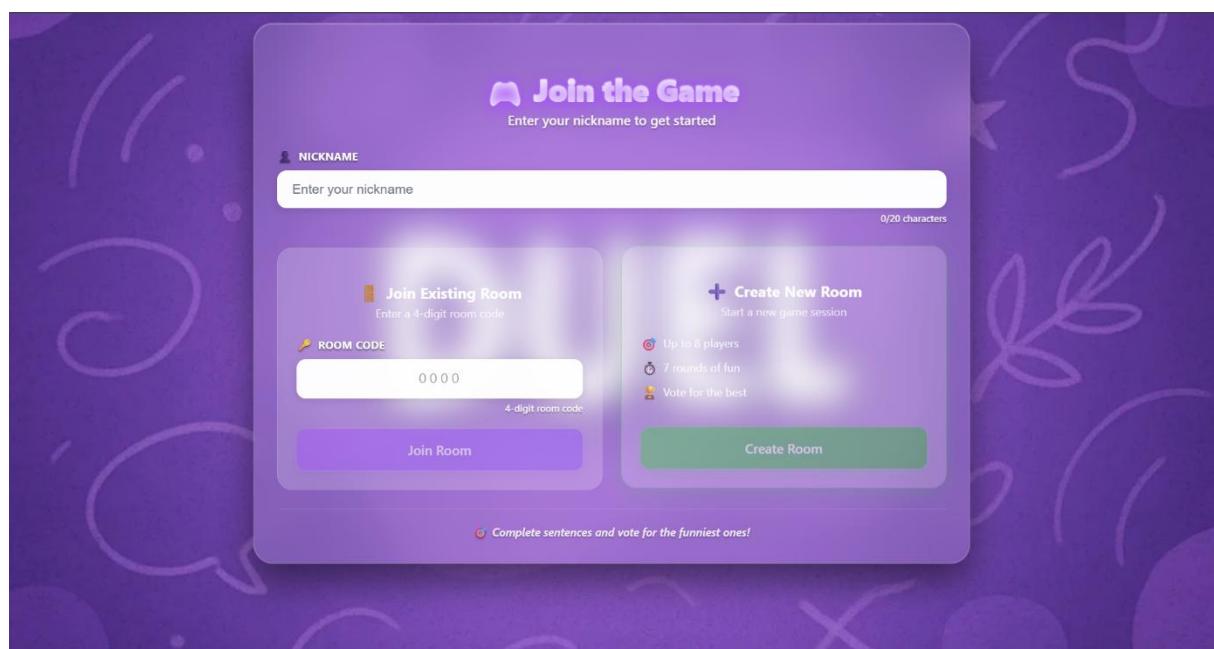
3. Abbildung von Präsentation

## 4.2 Ergebnis der Arbeit

### Ergebnis der Arbeit

Zu Beginn kann der Spieler einen Nickname und einen vierstelligen Raum Code eingeben, um entweder einem bestehenden Spiel beizutreten oder ein neues Spiel zu starten. Nach dem Beitritt wird automatisch ein zufällig ausgewählter Satzanfang (Starter) angezeigt. Der Spieler hat dann die Möglichkeit, diesen mit einer eigenen Fortsetzung zu vervollständigen. Die maximale Zeichenlänge beträgt 200 und die Eingabe wird validiert, sodass leere Texte oder zu lange Beiträge nicht abgeschickt werden können.

Sobald alle Spieler ihren Satz eingereicht haben, wird die Bewertungsphase gestartet. In dieser Phase kann jeder Spieler die eingereichten Sätze (ausser den eigenen) mit einem Emoji bewerten. Die Emojis haben unterschiedliche Punktwerte, die sich direkt auf das Scoring und das Ranking auswirken. Am Ende der Runde wird das Ranking inklusive der besten Sätze und der aktuellen Punkteverteilung angezeigt. Nach insgesamt sieben Runden erscheint automatisch eine Endauswertung mit einem Podium für die drei besten Spieler. Das Spiel läuft stabil, sowohl im Browser als auch mobil, und sorgt für ein unterhaltsames Multiplayer-Erlebnis.



4. Abbildung von dem Ergebnis

## 4.2.1 Kernfunktionalitäten

### Kernfunktionalitäten

Das Spiel verfügt über viele wichtige Funktionen, die ich nach und nach hinzugefügt habe. Am wichtigsten ist, dass man sich mit einem Raum Code einem Spiel anschliessen oder selbst einen neuen Raum erstellen kann. Jeder Raum hat eigene Spieler, Runden und einen Host, der das Spiel starten kann.

Zu Beginn jeder Runde wird automatisch ein zufälliger Satzanfang angezeigt. Diese Satzanfänge habe ich in einer Liste gespeichert, aus der das System dann einen zufällig auswählt. Die Spieler müssen diesen Satz dann weiterschreiben. Es gibt jedoch eine Eingabeprüfung, damit niemand etwas Leeres abschickt oder mehr als 200 Zeichen schreibt.

Anschliessend kommt das Voting-System, bei dem mit Emojis wie 😍, 😂, 🙄 oder 💩 für die Sätze abgestimmt werden kann. Jede Emoji gibt eine bestimmte Punktzahl, die direkt zusammengezählt wird. Daraus ergibt sich eine Rangfolge, die zeigt, wer wie viele Punkte hat – und diese wird live aktualisiert.

Bevor das Spiel startet, befinden sich alle Spieler: innen in einem Wartezimmer. Dort ist zu sehen, wie viele Spieler sich bereits im Spiel befinden. Wenn genug Spieler da sind, startet ein Countdown. Alternativ kann der Host auch direkt auf „Start“ drücken. Nach sieben Runden zeigt das Spiel eine Auswertung mit einem Podium, auf dem die drei besten Spieler angezeigt werden. Mit der Reset-Funktion kann im gleichen Raum noch einmal von vorne begonnen werden.

Ich habe darauf geachtet, dass alles auf dem Handy, dem Tablet und dem PC gut aussieht. Dafür habe ich ein responsives Design erstellt, das sich immer dem jeweiligen Gerät anpasst.

## 4.2.2 Technische Leistungen

### Technische Leistungen

Das Projekt verfügt über eine vollständig implementierte REST-API mit insgesamt zwölf Endpunkten, die alle zentralen Spielfunktionen wie Beitritt, Satzabgabe, Voting, Punktestand und Rundenfortschritt abdecken. Die Datenbankintegration wurde mithilfe von PostgreSQL umgesetzt. Dabei wurde auf performante und optimierte SQL-Abfragen geachtet, um schnelle Ladezeiten und eine saubere Datenverarbeitung sicherzustellen. Für die gesamte Codebasis kam TypeScript zum Einsatz, wodurch eine hohe Typsicherheit und eine bessere Wartbarkeit gewährleisteten sind. Das Deployment der Anwendung erfolgte über die Plattform Render. Dabei wurden sensible Daten wie Datenbank-URLs über Umgebungsvariablen verwaltet. Zusätzlich wurde das gesamte Projekt mit Git versioniert, inklusive strukturierter Commits, die eine klare Nachverfolgung der Entwicklungsschritte ermöglichen.

## 4.3 Funktionsbeschreibung

### 4.3.1 Komponenten

#### Komponenten

- **Join-Form:** Spielername + Raum Code Eingabe, Wechsellogik
- **Game Screen:** Anzeige eines zufälligen Satzanfangs + Eingabefeld für Satzende
- **Waiting-Room:** Warte-Raum für Spieler mit einem Host, der entscheiden kann, wann das Spiel startet – egal, ob der Timer fertig ist oder nicht
- **Zustandsverwaltung:** via React useState

### 4.3.2 Spielablauf

#### Spielablauf

1. **Spielstart und Raum-Beitritt:** Eingabe von Spielername und Raum Code
2. **Wartezimmer-Mechanik:** Anzeige aller beigetretenen Spieler mit Timer
3. **Spielrunde:** Zufällige Auswahl eines Satzanfangs, Eingabe der Beendigung
4. **Bewertungsphase:** Emoji-Voting für alle Sätze ausser dem eigenen
5. **Ranking:** Automatische Berechnung und Anzeige der Punkte
6. **Reset-Option:** Vollständiger Neustart des Spiels möglich

## 4.4 Arbeitsjournal

### 4.4.1 Tag 1: 02.06.2025

#### Tag 1: 02.06.2025

Heute haben wir mit der Abschlussarbeit begonnen.

Ich habe gelernt, wie TypeScript genau funktioniert, und mir wurde klar, wie ich mit React arbeite und die nötigen Installationen durchführe.

Ausserdem hatte ich Probleme mit den Styles und den Farben der Startseite.

Nachdem ich andere um ihre Meinung gebeten hatte, konnte ich schliesslich eine gute Lösung finden.

Heute ging es mir gut und ich war gut gelaunt.

### 4.4.2 Tag 2: 03.06.2025

#### Tag 2: 03.06.2025

Heute habe ich an meinem Abschlussprojekt weitergearbeitet.

Ich habe gelernt, wie man einen zufälligen Satz aus einer JSON-Datei ausliest und wie man in TypeScript einen beliebigen Satz auswählt und auch, wie man ein Eingabefeld erstellt, um die Endung der Sätze zu bestimmen.

Ausserdem hatte ich Schwierigkeiten mit dem Input-Feld und dem Styling, aber ich habe es so hinbekommen, dass es nicht schlecht aussieht.

Heute war ich gut gelaunt und es ging mir gut.

### 4.4.3 Tag 3: 04.06.2025

#### Tag 3: 04.06.2025

Heute habe ich an dem ersten Meilenstein weitergearbeitet und bin damit fast fertig geworden.

Ich habe gelernt, wie ich alle Antworten von Fake-Spielern anzeigen lassen kann und auch, wie man mit einem Effekt solche Ladepunkte erstellen kann.

Ausserdem hatte ich noch Probleme mit diesem Effekt, weil ich nicht wusste, wie ich ihn implementieren sollte. Mithilfe des Internets konnte ich dann aber eine gute Lösung finden.

Heute ging es mir gut und ich war gut gelaunt.

#### 4.4.4 Tag 4: 10.06.2025

##### **Tag 4: 10.06.2025**

Heute habe ich Meilenstein 1 fertiggestellt und mit Meilenstein 2 begonnen.

Ich habe gelernt, wie man ein Emoji-System hinzufügt und wie man es stylen kann, um alles kompakt und passend miteinander zu gestalten.

Ausserdem hatte ich Schwierigkeiten mit dem Styling, weil man den Next-Round-Button nicht sah und auch mit den Emojis bei den Auswahlen. Nachdem ich verschiedene Vorschläge aus dem Internet ausprobiert hatte, gelang es mir.

Heute war ich gut gelaunt und es ging mir gut.

#### 4.4.5 Tag 5: 11.06.2025

##### **Tag 5: 11.06.2025**

Heute habe ich mit Meilenstein 2 weitergemacht.

Ich habe gelernt, wie ich das Frontend mit dem Backend verbinde und welche Anpassungen ich vornehmen muss, damit alles reibungslos funktioniert.

Ausserdem hatte ich noch Probleme mit ein paar Erweiterungen, die ich währenddessen umsetzen musste, damit alles flüssig laufen würde. Am Ende des Tages funktionierte jedoch alles.

Mir ging es heute gut und ich war gut gelaunt.

#### 4.4.6 Tag 6: 16.06.2025

##### **Tag 6: 16.06.2025**

Heute bin ich fast mit Meilenstein 2 fertig geworden.

Ich habe gelernt, wie ich ein Ranking der besten Sätze erstelle und wie ich ein Podium erstelle, das die Spieler nach ihren besten Sätzen ranken.

Ausserdem hatte ich noch Schwierigkeiten, weil meine GET-sentence-Anfrage nicht mehr funktioniert hat. Nachdem ich der URL dann wieder richtig angepasst hatte, hat es funktioniert.

Heute war ich gut gelaunt und es ging mir gut.

#### 4.4.7 Tag 7: 17.06.2025

##### **Tag 7: 17.06.2025**

Heute habe ich Meilenstein 2 abgeschlossen und mit Meilenstein 3 weitergemacht.

Ich habe gelernt, wie ich meine Oberfläche benutzerfreundlicher gestalten und Animationen hinzufügen kann, damit alles realistischer wirkt.

Ausserdem hatte ich Probleme mit Positionen und Grössen, aber am Schluss konnte ich alles in Ordnung bringen.

Heute ging es mir gut und ich war gut gelaunt.

#### 4.4.8 Tag 8: 18.06.2025

##### **Tag 8: 18.06.2025**

Heute habe ich den Meilenstein 3 fast fertig gemacht.

Ich habe gelernt, wie man PostgreSQL benutzt und wie man auch die Installation durchführt und auch Voting, sowie Ranking System funktionsfähig gemacht.

Ausserdem hatte ich Schwierigkeiten, mit dem Voting und Ranking System aber als ich dann einen separaten Endpunkt machte, also einen Für die Sätze und einen für das Ranking von den Spielern, und dann ging es.

Heute war ich gut gelaunt und es ging mir gut.

#### 4.4.9 Tag 9: 20.06.2025

##### **Tag 9: 20.06.2025**

Heute gab es einen Ausfall der BMS, weshalb ich noch an meinem Abschlussprojekt arbeiten konnte.

Ich habe gelernt, wie ich ein Wartezimmer erstellen und eine Funktion einbinden kann, die es einem Host ermöglicht, den Start des Spiels zu bestimmen.

Ausserdem hatte ich Probleme, diese Force-Start-Funktion einzubinden, aber nachdem ich mein Backend und Frontend korrigiert hatte, funktionierte es.

Heute ging es mir gut und ich war gut gelaunt.

#### 4.4.10 Tag 10: 23.06.2025

##### Tag 10: 23.06.2025

Heute habe ich an der Veröffentlichung meines Spiels gearbeitet und zusätzlich die Option hinzugefügt, den Namen der beigetretenen Spieler zu sehen.

Ich habe gelernt, wie man das Deployment auf Render durchführt und wie man dort seine Datenbank aus PostgreSQL integriert.

Ausserdem hatte ich Schwierigkeiten, meinen Code anzupassen, da ich die URLs verbessern und auch ein paar Fetch-Funktionen anpassen musste.

Heute war ich gut gelaunt und es ging mir gut.

#### 4.4.11 Tag 11: 24.06.2025

##### Tag 11: 24.06.2025

Heute habe ich mit der Präsentation begonnen, sie fertiggestellt und einige Fehler behoben.

Ich habe gelernt, wie ich meinen Render-Server immer deployen kann, sobald mein GitHub-Repository aktualisiert ist, und auch, wie ich dies verwalten kann.

Ausserdem hatte ich Probleme mit meiner Dokumentation, weil ich nicht genau wusste, wie ich sie am besten gestalten sollte. Nachdem ich mal begonnen hatte, wurde mir klarer, wie es geht.

Heute ging es mir gut und ich war gut gelaunt.

#### 4.4.12 Tag 12: 25.06.2025

##### Tag 12: 25.06.2025

Heute habe ich meine Dokumentation fertiggestellt und einige Anpassungen in meinem Code vorgenommen.

Ich habe gelernt, wie man ein Inhaltsverzeichnis für Bilder in Word erstellt und wie man die Bilder so anpasst, dass sie dort angezeigt werden.

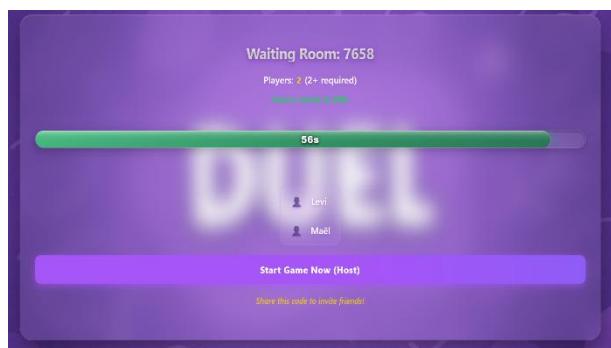
Am Anfang hatte ich noch Schwierigkeiten damit, zu verstehen, wie das geht, aber nachdem ich jemanden gefragt hatte, konnte ich es.

Heute war ich gut gelaunt und es ging mir gut.

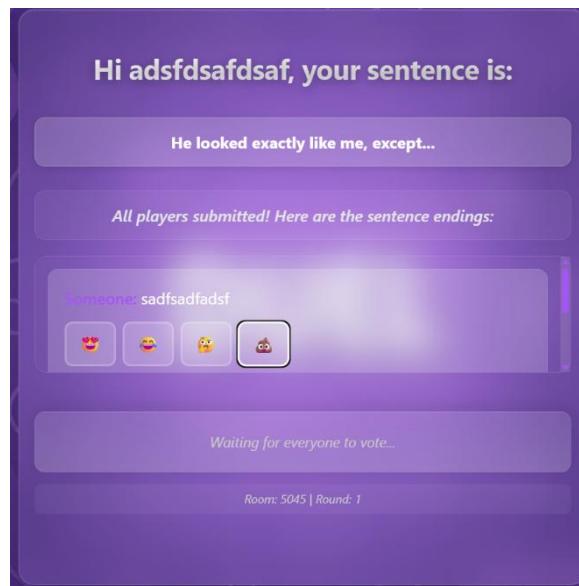
## 4.5 Testplan

### Testplan

Testfall	Beschreibung	Erwartetes Verhalten	Status
TF-001	Leeres Nickname-Feld	Fehlermeldung	✓
TF-002	Satzende länger als 200 Zeichen	Fehlermeldung	✓
TF-003	Satzende > 200 Zeichen	Eingabe wird abgeschnitten/Fehlermeldung	✓
TF-004	Satz korrekt gespeichert	Ausgabe in Konsole + Bestätigung	✓
TF-005	Emoji-Voting funktional	Vote wird gezählt und angezeigt	✓
TF-006	Eigener Satz nicht votbar	Voting-Buttons deaktiviert	✓
TF-007	Next Round Reset	Neuer Satzanfang + Input-Reset	✓
TF-008	Ranking korrekt	Spieler nach Score sortiert	✓
TF-009	Game Reset	Vollständiger State-Reset	✓
TF-010	Mobile Responsiveness	Layout passt sich an	✓
TF-011	Wartezimmer Timer	60-Sekunden-Countdown funktioniert	✓
TF-012	Host Force-Start	Host kann Spiel manuell starten	✓
TF-013	Datenbankabfrage in pgAdmin 4	< 100ms	✓



6. Abbildung von TF-011 und TF-012



5. Abbildung von TF-005

## 4.6 Herausforderungen und Planänderungen

### Herausforderungen und Planänderungen

Während des Projekts gab es verschiedene Herausforderungen zu lösen. Eine der grössten war das Emoji-Voting-System, da mir anfangs nicht klar war, wie die Emojis richtig gespeichert und die Punkte vergeben werden. Ich musste lernen, wie man Votes mit einer Datenbank verbindet und wie man sie dann richtig zusammenzählt.

Auch die Verbindung von Frontend und Backend bereitete mir anfangs Schwierigkeiten. Ich hatte viele Fetch-Fehler und musste die API-Endpunkte mehrfach anpassen, damit alles funktionierte. Insbesondere das Ranking-System bereitete Probleme, da es zu Konflikten mit den Sätzen und den Votes kam. Später habe ich dann zwei separate Endpunkte erstellt – einen für die Sätze und einen für das Ranking –, wodurch es besser funktionierte.

Beim Design hatte ich Probleme, alles auf dem Handy gut aussehen zu lassen. Ich musste viel mit CSS herumprobieren, bis alles überall schön und übersichtlich war. Schliesslich habe ich auf „Mobile-First“ geachtet und mit flexiblen Layouts gearbeitet.

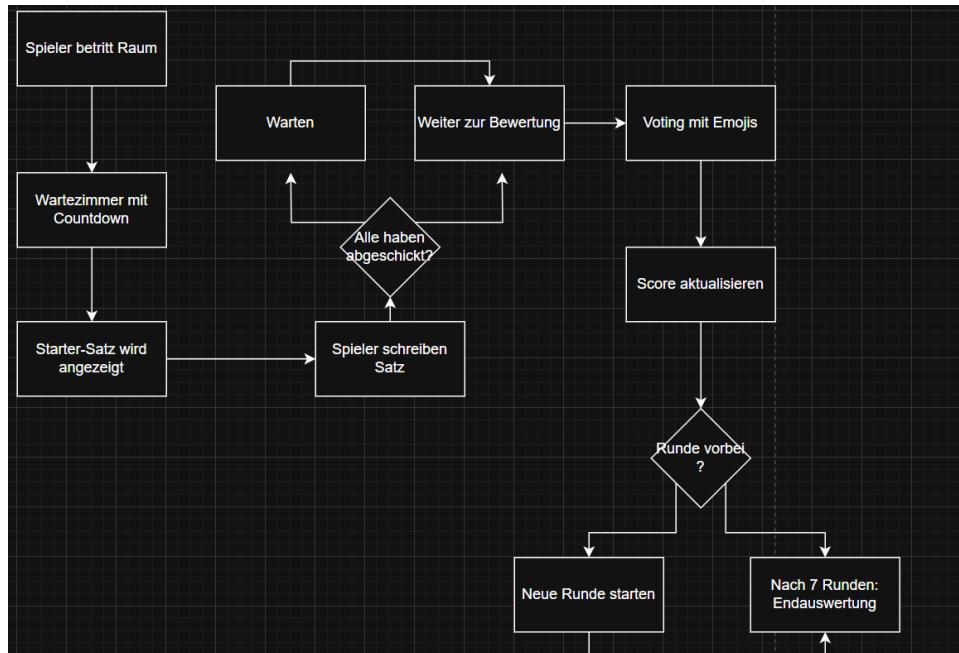
Beim Deployment auf Render gab es auch einige Schwierigkeiten, vor allem mit den Umgebungsvariablen und der Datenbankverbindung. Ich musste das Setup mehrmals testen, bis alles richtig lief. Am Schluss war das Ergebnis aber gut.

Ursprünglich war auch geplant, Web Sockets einzubauen, damit alles noch mehr in Echtzeit läuft. Ich habe jedoch gemerkt, dass dies den zeitlichen Rahmen sprengen würde. Darum habe ich das auf später verschoben und mich auf die Kernfunktionen konzentriert. Das war eine gute Entscheidung, denn so konnte ich das Projekt sauber abschliessen.

Insgesamt habe ich gelernt, dass man nicht alles auf einmal machen kann und dass es wichtig ist, zuerst die wichtigen Dinge umzusetzen. Bei Problemen habe ich nie aufgegeben, sondern immer nach einer Lösung gesucht.

## 4.7 Flussdiagramm

### Flussdiagramm



7. Abbildung vom Flussdiagramm

## 5. Anhang

### 5.1 Glossar

- **Backend:** Serverseitige Anwendung auf Basis von Node.js und Express, verwaltet Spiellogik und Kommunikation mit der Datenbank.
- **Frontend:** React-Anwendung, die das Spielerlebnis im Browser bereitstellt.
- **PostgreSQL:** Relationale Datenbank zur Speicherung von Spielern, Räumen, Sätzen und Stimmen.
- **pg:** Node.js-Bibliothek zur Anbindung an PostgreSQL-Datenbanken.
- **API:** Schnittstelle zwischen Frontend und Backend mittels REST-HTTP-Endpunkten.
- **Room:** Ein virtueller Spielraum mit eigenem 4-stelligem Code. Speichert Host, Teilnehmer, Spielstatus und Rundenfortschritt.
- **Player:** Ein Mitspieler im Spielraum, identifiziert durch einen Nicknamen.
- **Starter:** Ein zufällig ausgewählter Satzanfang, der als Schreibanreiz dient.

- **Sentence:** Vom Spieler verfasste Fortsetzung zum Starter, wird pro Runde eingereicht.
- **Vote:** Bewertung einer Sentence durch Emojis:
  - 😍 = 3 Punkte
  - 😂 = 2 Punkte
  - 😢 = 1 Punkt
  - 💩 = 0 Punkte
- **Round:** Eine Spielrunde – es gibt maximal 7. Jede Runde besteht aus Schreiben, Abstimmen und Auswertung.
- **Countdown:** 60-sekündiger Timer, der den Beginn einer Runde signalisiert.
- **Ranking:** Spielerplatzierung basierend auf den gesammelten Punkten aus allen Runden.
- **Join-Form:** Startformular zum Beitreten oder Erstellen eines Spielraums.
- **Waiting-Room:** Lobby-Ansicht vor Spielbeginn, zeigt Spieler und Countdown.
- **Game Screen:** Hauptspielloberfläche mit Schreibfeld, Voting System, Zwischen- und Endauswertungen.
- **joinRoom(roomCode, nickname):** Spieler einem Raum hinzufügen.
- **createRoom(nickname):** Neuen Raum mit Host erstellen.
- **submitSentence(roomCode, nickname, sentence):** Satz in aktueller Runde speichern.
- **vote(roomCode, sentenceId, nickname, emoji):** Satz bewerten.
- **getSentences(roomCode):** Alle Sätze der aktuellen Runde abrufen.
- **getBestSentences(roomCode):** Bestbewertete Sätze aus allen Runden laden.
- **advanceRound(roomCode):** Nächste Runde starten.
- **getRound(roomCode):** Aktuelle Rundenzahl abrufen.
- **updateScores(roomCode):** Punkte aktualisieren nach Abstimmung.
- **getRanking(roomCode):** Gesamtrangliste abrufen.
- **startGameCountdown(roomCode):** Countdown manuell starten (Host).

- **forceStartGame(roomCode, nickname)**: Countdown zwangsweise starten (Host).
- **isHost(roomCode, nickname)**: Prüft, ob ein Spieler der Host ist.
- **getPlayers(roomCode)**: Liste der Spieler im Raum abrufen.

## 5.2 Codeausschnitte

### Codeausschnitte

#### 5.2.1 Emoji-Voting System

##### Emoji-Voting System

```
const emojis = ["😊", "😎", "🥳", "💩"]  
const emojiPoints: { [emoji: string]: number } = {  
    "😊": 3,  
    "😎": 2,  
    "🥳": 1,  
    "💩": 0,  
}  
  
const handleVote = async (sentenceId: string, emoji: string) => {  
    setUserVotes((prev) => {  
        const updated = { ...prev }  
        if (updated[sentenceId] === emoji) {  
            delete updated[sentenceId]  
        } else {  
            updated[sentenceId] = emoji  
        }  
        return updated  
    })  
  
    try {  
        await vote(roomCode, sentenceId, nickname, emoji)  
    } catch (error) {  
        console.error("Vote failed:", error)  
        alert("Voting failed.")  
        setUserVotes((prev) => {  
            const updated = { ...prev }  
            delete updated[sentenceId]  
            return updated  
        })  
    }  
}
```

8. Zeigt das Emoji-Voting-System mit der Punktevergabe

## 5.2.2 Polling-Mechanismus für Echtzeit-Updates

### Polling-Mechanismus für Echtzeit-Updates

```
useEffect(() => {
  const interval = setInterval(async () => {
    try {
      const currentRound = await getRound(roomCode)
      if (currentRound !== round) {
        setRound(currentRound)
        await handleNewRound()
      }

      if (showAllSentences && !allVoted) {
        const res = await hasEveryoneVoted(roomCode)
        if (res.allVoted) {
          setAllVoted(true)
          await updateScores(roomCode)

          const ranking = await getRanking(roomCode)
          const newScores: Record<string, number> = {}
          ranking.forEach((player: { nickname: string; total_points: number }) => {
            newScores[player.nickname] = player.total_points
          })
          setTotalScores(newScores)
        }
      }
    } catch (error) {
      console.error("Polling error:", error)
    }
  }, 1000)
}, [round, showAllSentences, allVoted, updateScores])
```

9. Dargestellt ist der Polling-Mechanismus, der für Echtzeit-Updates sorgt

## 5.2.3 Datenbankabfrage für Sätze mit Votes

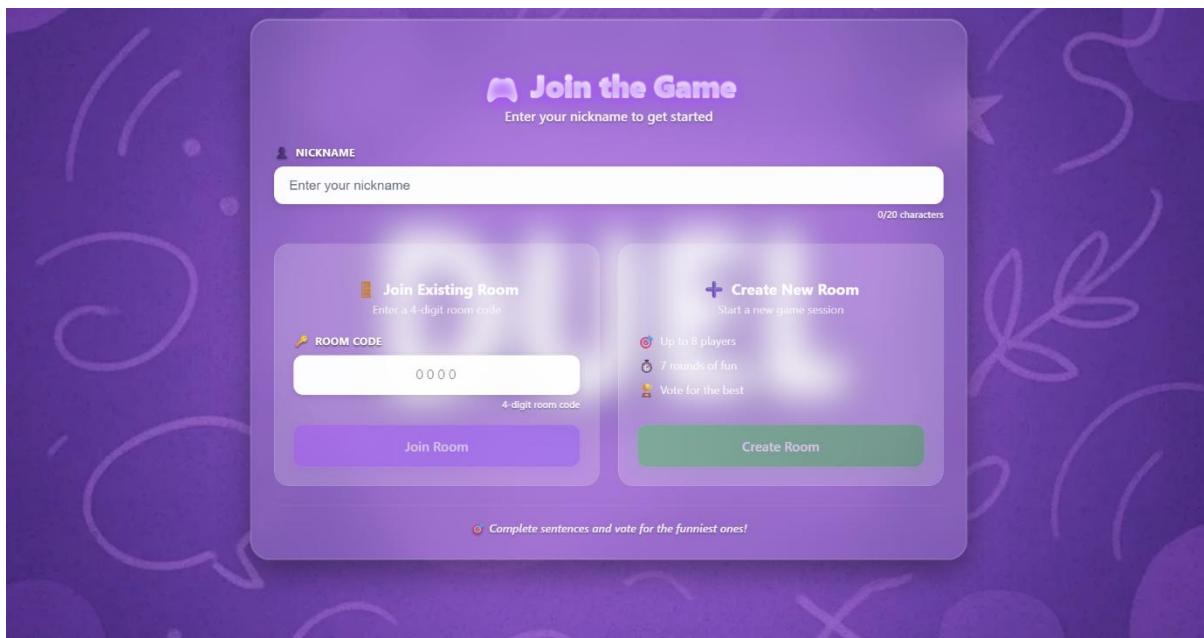
### Datenbankabfrage für Sätze mit Votes

```
async function getSentences(roomCode) {
  const round = await getRound(roomCode);
  const result = await pool.query(
    `SELECT s.id, s.text, p.nickname AS author, s.round, s.room_code,
       ARRAY_REMOVE(ARRAY_AGG(v.emoji), NULL) AS votes
    FROM sentences s
    JOIN players p ON p.id = s.author_id
    LEFT JOIN votes v ON v.sentence_id = s.id
    WHERE s.room_code = $1 AND s.round = $2
    GROUP BY s.id, p.nickname
    ORDER BY s.id,
    [roomCode, round]
  );
  return result.rows;
}
```

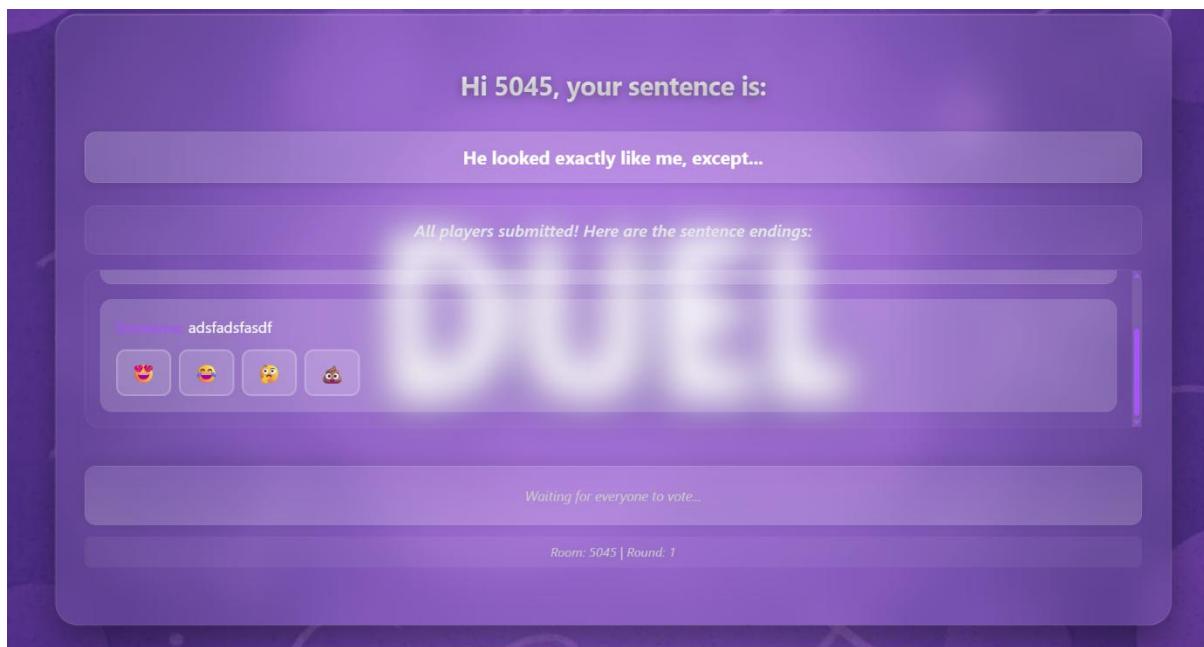
10. Präsentiert eine komplexe SQL-Abfrage

## 5.3 Screenshot-App

### Screenshot-App



11. Abbildung von Startseite der App



12. Abbildung von Game-Screen

## 5.4 KI-Chat-Auszüge

### KI-Chat-Auszüge

In den KI-Chat-Auszügen wurden verschiedene technische Themen behandelt. Unter anderem wurde Hilfe bei der Definition von TypeScript-Typen angeboten und die Optimierung von PostgreSQL-Abfragen diskutiert. Auch Probleme mit dem CSS-Flexbox-Layout wurden diskutiert. Darüber hinaus wurde Unterstützung bei der richtigen Verwendung von Dependencies im React-Hook useEffect angeboten. Ein weiteres Thema war die Konfiguration für das Deployment auf der Plattform Render.

## 5.5 Quellen und Literatur

### Quellen und Literatur

- React Documentation - <https://react.dev/>
- TypeScript Handbook - <https://www.typescriptlang.org/docs/>
- PostgreSQL Documentation - <https://www.postgresql.org/docs/>
- Chatgpt - <https://chatgpt.com/>
- Claude AI - <https://claude.ai/>
- Deepl-Write - <https://www.deepl.com/de>
- Rechtschreibpruefung24.de - <https://rechtschreibpruefung24.de>

## 5.6 Bildverzeichnis

### Bildverzeichnis

1. Abbildung von Roadmap .....	6
2. Deployement Test vom Backend in Render .....	7
3. Abbildung von Präsentation.....	9
4. Abbildung von dem Ergebnis .....	10
5. Abbildung von TF-005 .....	17
6. Abbildung von TF-011 und TF-012 .....	17
7. Abbildung vom Flussdiagramm.....	19
8. Zeigt das Emoji-Voting-System mit der Punktevergabe.....	21
9. Dargestellt ist der Polling-Mechanismus, der für Echtzeit-Updates sorgt .....	22
10. Präsentiert eine komplexe SQL-Abfrage.....	22
11. Abbildung von Startseite der App .....	23
12. Abbildung von Game-Screen .....	23
13. Abbildung von der Checkliste .....	25
14. Abbildung von Github Repo .....	25

## 5.7 Checklist

### Checkliste

Checkliste Individuelles Abschlussprojekt Dokumentation 2025		Bemerkungen
<i>v1.2 - Kastell - 23.5.2024</i>		
<b>Wichtige Hinweise</b>	Dokumentation ist für Fachpersonen verständlich Eigenleistung und Unterstützungen sind klar deklariert	<input checked="" type="checkbox"/> <input type="checkbox"/>
<b>Allgemein</b>	Kopfzeile Projektname Titel Fusszeile Datum, Version, AutorIn, Seitenzahl Seitenlayout: Keine Überlappung Seitenränder, Quer/Hochformat Beschriftung der Bilder/Grafiken Einsatz von aussagekräftiger Grafiken (Netzwerkplan, DB Schema)	<input checked="" type="checkbox"/>
<b>Titelblatt</b>	Klar und übersichtlich gestaltet Überbegriff: Individuelle Abschlussprojekt BLJ Projektname (aussagekräftig / max 20 Zeichen) Name, Abgabedatum, Name der Lehrfirma Version des Dokuments	<input checked="" type="checkbox"/>
<b>Inhaltsverzeichnis</b>	Kapitel nummeriert (z.B.: 2.1, 2.1.1 usw.) Seitenzahlen korrekt angegeben Formatierung überprüft	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
<b>Einleitung</b>	Änderungstabelle/Versionierung (tabellenform) Aufgabenstellung und Projektbeschreibung Mögliche Risiken vor Projektbeginn	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
<b>Planung</b>	Terminplan (Gantt-Diagramm oder Screenshot Github Project) vorhanden Entscheidungswege und Möglichkeiten (Entscheidungsmatrix)	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
<b>Hauptteil</b>	Detaillierte Beschreibung des Vorgehens und der Zwischenschritte Ergebnisse der Arbeit Entscheidungen sind formuliert Arbeitsjournal vorhanden Testplan/Testfälle mit Ergebnissen Personliches Fazit	<input checked="" type="checkbox"/> Ich habe dies einfach bei Reflexion und Fazit am Schluss
<b>Anhang</b>	Quellenangaben und Literaturverzeichnis vorhanden Glossar/Begriffserklärungen vorhanden (Github / Dokument) Bildverzeichnis vorhanden Programm-Code, Scripts, Foto-Dokumentation (Github / Dokument) Relevante AI-Chat-Prompts/Auszüge Testplan/Testfälle (optional) Ausgefüllte Checkliste	<input checked="" type="checkbox"/> Ich wusste nicht genau was Foto-Dokumentation ist also habe ich einfach Screenshots von meiner App gemacht
<b>Code/Konfigurationen</b> <i>(Dateien bevorzugt auf Github)</i>	Link zum Github Repository vorhanden Programm-Code folgt Clean-Code Richtlinien Wichtige Module, Klassen, Funktionen sind kommentiert Aussagekräftige Namen für Dateien, Klassen, Funktionen, DB Felder, ... Programm-Dateien Header mit Autor, Datum, Version, Beschreibung	<input checked="" type="checkbox"/>

13. Abbildung von der Checkliste

## 5.8 Github sowie Spiel-Link

### Github sowie Spiel-Link

Github: [Github](#) und Spiel-Link: [Finish-This! Spiel](#)

14. Abbildung von Github Repo

## 6. Fazit und Reflexion

### Fazit und Reflexion

Das Projekt „Finish This!“ war für mich ein grosser Erfolg. Ich konnte alle geplanten Funktionen umsetzen und sogar noch Extras wie ein Reset-System und Designverbesserungen hinzufügen. Die App ist online, läuft stabil und macht im Multiplayer-Modus richtig Spass.

Technisch habe ich viel gelernt: TypeScript hat zu weniger Fehlern geführt und mithilfe von React konnte ich den Code besser strukturieren. Die Datenbank mit PostgreSQL war zuverlässig und schnell, und das Deployment über Render war einfach und praktisch.

Besonders stolz bin ich darauf, dass ich schwierige Dinge wie den Spielstatus (State) gut im Griff hatte und die Kommunikation mit dem Backend reibungslos funktionierte. Auch das Design funktioniert jetzt auf allen Geräten gut.

Ich habe in diesem Projekt sehr viel gelernt – über moderne Web-Technologien, Datenbanken, Backend-Entwicklung und darüber, wie man ein solches Projekt gut plant. Auch die Dokumentation und das Testen waren wichtig, um den Überblick zu behalten.

Für das nächste Mal würde ich früher mit automatisierten Tests beginnen, eventuell auch Web Sockets einbauen und mehr auf den Benutzer-Login und die Performance achten.

Insgesamt hat mir das Projekt grossen Spass gemacht. Es war grossartig, eine eigene Idee umzusetzen und zu sehen, wie alles zusammenpasst.