Pflichtenheft

<Projektname> Plattform zum Vergleich von Spiele-KIs

<Projektnummer> Gruppe 2

Änderungshistorie				
Version	**Datum**	**Kapitel**	**Änderung**	**Name**
0.1	25.04.2024	Alle	Anlegen und Füllen	Justine Buß
0.2	29.04.2024	3	Formulieren	Justine Buß, Maximilian Bachmann
0.3	30.04.2024	5, 6, 7, 8.5	Formulieren	Justine Buß, Maximilian Bachmann
1.0	30.04.2024	Alle	Links, Diagramme, Überarbeiten	Justine Buß
1.1	02.05.2024	Alle	Nachbessern	Justine Buß
1.2	05.05.2024	3, 6	Überarbeiten, Diagramme	Justine Buß, Alexander Roos
1.3	06.05.2024	3, 7, 8	Überarbeiten, Diagramme	Justine Buß, Sven Reinhard
1.4	07.05.2024	4	Mockups, Formulieren	Omar Karkotli
2.0	07.05.2024	Alle	Nachbessern	Justine Buß

Pflichtenheft

Herausgeber	Technische Hochschule Mittelhessen – FB06 Mathematik, Naturwissenschaften und Informatik	
Dateiname	Pflichtenheft_ver_1.4	
Dokumentenbezeichnung	Pflichtenheft: Plattform zum Vergleich von Spiele-KIs	
Version	1.4	
Stand	Dienstag, 7. Mai 2024	
Status	In Bearbeitung	
Autoren	Justine Buß, Thorben Jones, Alexander Roos, Maximilian Bachmann, Omar Karkotli, Sven Reinhard, Pascal Waldschmidt	
Freigegeben von		
Ansprechpartner	Justine Buß	justine.buss@mni.thm.de
Kurzinfo	„Technische Hochschule Mittelhessen Softwaretechnikprojekt. Pflichtenheft“	

Pflichtenheft

Inhaltsverzeichnis

Pflichtenheft

Abbildungssverzeichnis

Tabellenverzeichnis

Pflichtenheft

1 Einleitung

Das vorliegende Pflichtenheft enthält die an das zu entwickelnde Projekt gestellten funktionalen sowie qualitativen Anforderungen. Mit diesen werden die Rahmenbedingungen für die Entwicklung festgelegt, die im Pflichtenheft detailliert ausgestaltet werden.

2 Allgemeines

2.1 Ausgangssituation

In Anbetracht der wachsenden Bedeutung künstlicher Intelligenz in verschiedenen gesellschaftlichen Bereichen ist es zunehmend erforderlich, ein tieferes Verständnis für KI-Konzepte zu entwickeln und diese anzuwenden. Insbesondere im Bildungsumfeld wird es als entscheidend erachtet, Studierenden praxisnahe Erfahrungen zu ermöglichen, um ihre Fähigkeiten in der KI-Entwicklung zu vertiefen. Vor diesem Hintergrund gewinnt das Konzept des Lernens und Lehrens mit spielerischen Methoden zunehmend an Bedeutung. Der Einsatz von interaktiven Plattformen und erlebnisorientierten Ansätzen bietet eine effektive Möglichkeit, das Verständnis und die Anwendung von KI-Konzepten zu fördern. Durch die Integration dynamischer Elemente in den Lernprozess können Interessierte und Studierende auf unterhaltsame Weise praxisnahe Erfahrungen sammeln und ihre Fähigkeiten in der KI-Entwicklung auf ansprechende Art und Weise vertiefen. Diese Herangehensweise erleichtert nicht nur das Verständnis komplexer Konzepte, sondern trägt auch dazu bei, das Interesse und die Motivation der Lernenden zu steigern, da sie aktiv am Prozess beteiligt sind und direktes Feedback erhalten.

Um diesen Bedarf zu decken, ist es entscheidend, eine benutzerfreundliche Plattform zu schaffen, die das Verhalten von KIs durch interaktive Erfahrungen erkundet, verschiedene KIs vergleicht und das Testen eigener KIs ermöglicht. Während bereits existierende Plattformen einzelne Spiele mit KI-Unterstützung anbieten, fehlt es bisher an einer ganzheitlichen Lösung, die mehrere Spiele integriert und es den Nutzern ermöglicht, sowohl gegen andere Nutzer und verschiedene KIs anzutreten als auch eigene KI-Entwicklungen hochzuladen und zu testen.

Pflichtenheft

2.1 Projektbezug

Das Softwaretechnikprojekt an der Technischen Hochschule Mittelhessen unter der Leitung von Prof. Dr. Frank Kammer [2] strebt mit der Entwicklung einer Plattform, die den Benutzern eine interaktive Möglichkeit zum Sammeln praxisnaher Erfahrungen ermöglicht, die Schließung genau dieser Lücke an. Diese Plattform soll es den Nutzern ermöglichen, nicht nur gegen andere Nutzer anzutreten, sondern auch gegen verschiedene KIs zu spielen. Dabei ist es entscheidend, dass die Plattform benutzerfreundlich gestaltet ist und eine Vielzahl von Spielen unterstützt, um den Lernprozess im Bereich der KI-Entwicklung effektiv und ansprechend zu gestalten.

Anzuwendende Technologien:

- Programmiersprache: Python [3]
- Softwareplattform: Docker [4]
- Maschinelles Lernen Framework: TensorFlow [5] mit Keras [6]
- KI-Entwicklungsframework: AlphaZero [7, 11]

2.4 Beteiligte

Rolle	Name	Fachbereich/ Studienfach	Kontaktinformation
Projektleitung	Thorben Jones	MNI/Ingenieur-Informatik	thorben.jones@mni.thm.de
Stellvertretende Projektleitung	Justine Buß	MNI/Ingenieur-Informatik	justine.buss@mni.thm.de
Teammitglied	Alexander Roos	MNI/Ingenieur-Informatik	alexander.roos@mni.thm.de
Teammitglied	Maximilian Bachmann	MNI/Informatik	maximilian.lars.bachmann@mni.thm.de
Teammitglied	Omar Karkotli	MNI/Informatik	omar.karkotli@mni.thm.de
Teammitglied	Sven Reinhard	MNI/Informatik	sven.roman.reinhard@mni.thm.de
Teammitglied	Pascal Waldschmidt	MNI/Informatik	pascal.waldschmidt@mni.thm.de
Dozent	Prof. Dr. Frank Kammer	MNI	frank.kammer@mni.thm.de

Tabelle 1: Interne Beteiligung

2.5 Zielgruppen

Die Zielgruppen für unsere Plattform sind vielfältig. Zum einen richten es sich an Studierende und Dozenten der Hochschule, die in verschiedenen Modulen den Einsatz von KI kennenlernen, lehren und ihre eigenen KIs auf der Plattform testen können. Dadurch soll eine Lern- und Lehrunterstützung geboten und ein Einblick in vortrainierte KI-Lösungen ermöglicht werden. Zum anderen richtet sich das Projekt an Entwickler, die ihre Kenntnisse und Fähigkeiten im Bereich der KI-Entwicklung erweitern möchten und neue Anwendungen erforschen wollen.

Schließlich spricht die Plattform alle KI-Interessenten und Enthusiasten an, die mehr über die Entwicklung und Anwendung von KI erfahren möchten, sowie eigene KIs testen und verbessern wollen.

3 Implementierungsentwurf

3.1 Gesamtsystem

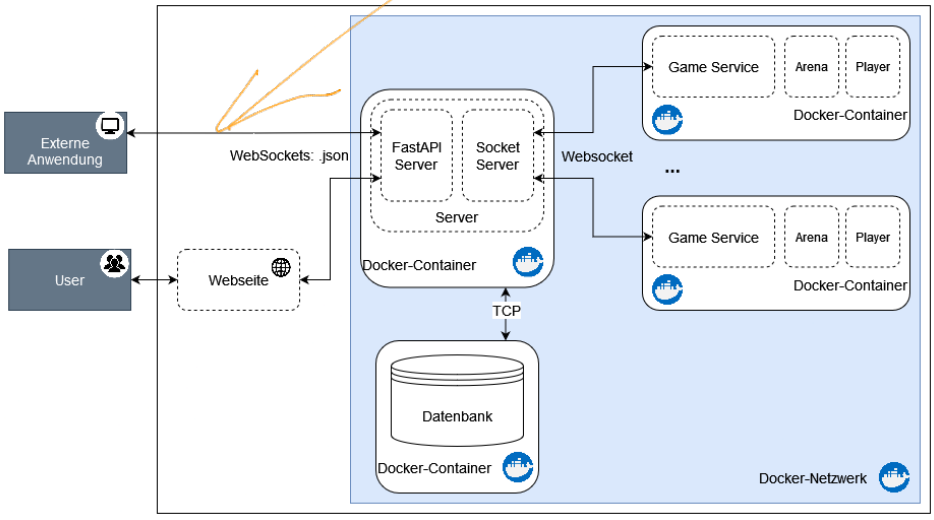

Abbildung 1: Architekturentwurf Gesamtsystem

Im Rahmen des Implementierungsentwurfs wird die Gesamtarchitektur des Systems umfassend beschrieben. Dies umfasst die verschiedenen Systemkomponenten sowie deren Interaktionen, Interdependenzen und Schnittstellen. Das Ziel dieses Abschnitts ist es, einen klaren Überblick über die geplante Softwarelösung zu geben und die grundlegenden Designentscheidungen zu erläutern. Dabei wird insbesondere auf die Integration der Komponenten und die Struktur des Gesamtsystems eingegangen.

Die Architektur des Systems basiert auf dem Client-Server Prinzip ergänzt mit Microservice-Ansätzen. Die Hauptkomponente stellt ein Serverkonzept dar, der als Vermittler zwischen den Clients (einer Webseite und externen Anwendungen) und den dahinterliegenden Game-Clients fungiert. Die Clients kommunizieren über WebSockets [19] mit dem Webserver, der Anfragen an die entsprechenden Docker-Container weiterleitet. Die Docker-Container stellen die verschiedenen Spielfunktionen bereit und können je nach Anforderung skaliert werden. Der Implementierungsentwurf fokussiert sich darauf, wie der Server die Interaktionen zwischen den Clients und den Docker-Containern verwaltet und wie die Komponenten nahtlos integriert sind, um die gewünschte Softwarelösung effizient zu realisieren.

3.2 WebSocket mit FastAPI

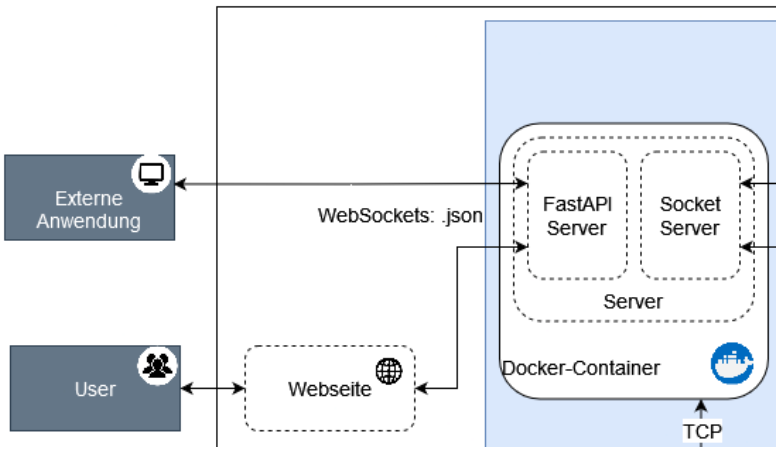

Abbildung 2: Architekturentwurfsausschnitt WebSocket Kommunikation

Die Kommunikation zwischen dem Server und der Webseite oder der externen Anwendung (User Clients) findet über das Kommunikationsprotokoll WebSocket statt. Es bietet eine bidirektionale und persistente Verbindung zwischen dem Client und dem Server und muss im Gegensatz zu HTTP, das auf einer Anfrage-Antwort-Struktur beruht, nicht für jede Interaktion eine neue Anfrage stellen. So lassen sich nach einem erfolgreichen Handshake (Protokoll, das auf TCP basiert) Nachrichten in Echtzeit über eine aufrechterhaltene Verbindung senden.

Zur Integration von WebSocket in den Python basierten Server wird das Framework FastAPI [8] implementiert. Dieses unterstützt die Integration von WebSocket-Endpunkten und WebSocket-Instanzen, über die Nachrichten asynchron empfangen und gesendet werden können. Die bereitgestellten Endpunkte ermöglichen die Interaktion der User mit den Spielfunktionen, wie mit dem Server verbinden, Erstellen einer neuen Lobby, konfigurieren und starten eines Spiels, sowie das letztendliche Spielen eines Spiels durch gemachte oder zurückgenommene Züge. Der Server antwortet über diese Verbindung mit Antwortzügen, dem in PyGame [9] dargestellten aktualisierten Spielfeld oder Verbindungsantworten.

Die Übertragung der Daten findet, abgesehen von den übertragenen Spielbrettern, im JSON Format statt. Die Nutzung ist für WebSockets leicht handhabbar und einfach zu implementieren. Es ermöglicht zudem die Spielinformationen kompakt und für den Menschen leserlich zu speichern. Übertragen werden hier somit: Lobby Tokens, die zur eindeutigen Identifizierung der Räume verwendet werden, Zug- und Spielbrettinformationen, sowie Serveranfragen für Datenaustausch.

3.3 Server in Python

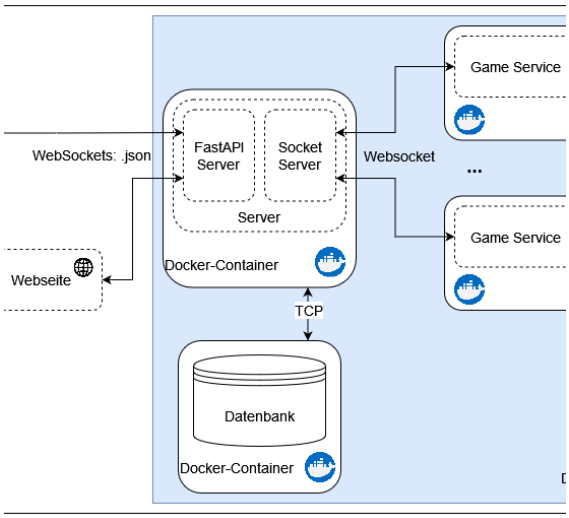

Abbildung 3: Architekturentwurfsausschnitt Server

Die Hauptkomponente des Systems, die die gesamte Kommunikation steuert, ist ein in einem Docker-Container ausgeführter Server, der in Python implementiert ist. Er dient als zentraler Ansprechpartner für alle Anfragen und seine Hauptaufgabe besteht darin, den Nachrichtenfluss zwischen den verschiedenen Teilen des Systems zu koordinieren, das System zu steuern, die Verbindungen aufzubauen und die verschiedenen Räume, Benutzer, Tokens und Docker-Container zu verwalten.

Der Server besteht aus zwei Teilservern, die aufgrund der unterschiedlichen Anbindungen an die User Seite und die Spielinstanzen in Docker verschieden implementiert sind.

Der FastAPI Server stellt die Schnittstelle zu den WebSocket Verbindungen, also den User Clients, dar. Er verwaltet alle Anfragen, die von der Benutzer Seite an den Server herangetragen werden und welche an die Benutzer wieder zurückgeleitet werden müssen. Dies umfasst Verbindungsanfragen von neuen und bestehenden Nutzern sowie die Verwaltung und Erzeugung von Benutzer- und Lobby-Tokens durch Hashing des aktuellen Zeitstempels mit dem SHA-256-Verfahren.

Der zweite Teil umfasst einen Socket [10] Server, der die Schnittstelle zu den Game Clients darstellt. Die Kommunikation findet hier über Sockets statt, mit Verwendung der WebSocket Bibliothek zur besseren und einfacheren Handhabung. Entsprechend der Anfragen auf der User-Client Seite werden bei neu erzeugten Spielen Docker-Container mit entsprechenden Spielinstanzen erzeugt und von dem Socket Server verwaltet. Dabei sind die Docker-Container eindeutig identifizierbar und lassen sich somit den Lobby Tokens zuordnen.

Die Verwaltung und Speicherung der Testdaten der vortrainierten KI-Implementierungen findet auf einer Datenbank in einem eigenen Docker-Container statt. Die Kommunikation wird von der Serverseite aus über TCP gesteuert. So sind alle Testdaten an einem Ort hinterlegt und können beim Erstellen einer neuen Spielinstanz in eigenen Docker-Container geladen werden. Somit muss nicht jede Spielinstanz alle Testdaten der KI-Implementierungen kennen oder speichern.

3.4 Dockerbasierte Spielinstanzen

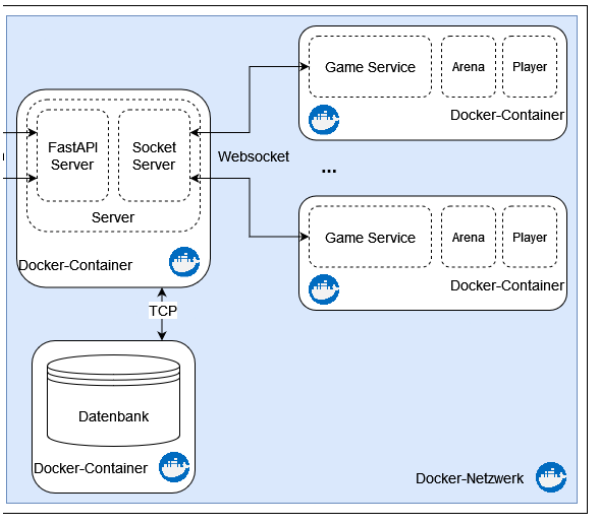

Abbildung 4: Architekturentwurfsausschnitt Docker-Container

Zur Bereitstellung eines Mehrspielerbetriebs und der Unterstützung zukünftiger Skalierbarkeit werden die Spielinstanzen in separaten Docker-Containern ausgeführt. Jede Spielinstanz kommuniziert dabei über den Websocket mit dem zentralen Server. Bei dessen Start meldet sich jede Spielinstanz als Game Client am Server an. Wird eine neue Instanz erstellt, generiert der Server einen eindeutigen Schlüssel, der diesem Docker-Container zugeordnet wird. Bei zukünftiger Kommunikation verwaltet der Server diese Schlüssel und leitet Anfragen entsprechend an die Docker-Container weiter oder von den Containern zu den entsprechenden Clients. Innerhalb der Docker-Container werden die Spielinstanzen mit Python und den entsprechenden Bibliotheken, die für die Ausführung des jeweiligen Spiels erforderlich sind, implementiert.

Zum Verwalten der Container für die Spielinstanzen (SI-Container) wird der Server genutzt. Dieser nutzt das Docker SDK für Python [20], um eine Verbindung zwischen dem Server innerhalb des Containers zu der Docker-Engine, welche alle Container verwaltet, herzustellen. Über diese Verbindung wird der Server den LifeCycle der SI-Container verwalten. Das Docker SDK ermöglicht außerdem das Monitoring der SI-Container.

Beim Erzeugen der Container greift der Server auf die entsprechend benötigten vortrainierten KI-Implementierungen zu. Das Trainieren der KIs erfolgt auf einem Unsuperviced Learning Paradigma, dem Reinforcement Learning. Der Ansatz orientiert sich an der AlphaZero [7, 11] Lösung von Google DeepMind und spezifischer auf einem angepassten alphazero framework [12]. Die Python Bibliotheken Keras und TensorFlow (sowie davon abhängige Bibliotheken) und PyTorch [15] unterstützen das Trainieren der KIs in neuralen Netzen.

4 Webseite

Für die Erstellung des Frontends kommen Vue.js Version 3.4.27 [17] und Bootstrap Version 5.3.3 [18] zum Einsatz. Vue.js ist eine JavaScript-Bibliothek, die besonders für die Entwicklung interaktiver Webanwendungen und Single-Page-Anwendungen geeignet ist. Ihr komponentenbasiertes Modell ermöglicht es, die Benutzeroberfläche in wiederverwendbare Komponenten zu zerlegen. Diese Komponenten können dann leicht gewartet und skaliert werden, was die Entwicklung effizienter und flexibler macht.

Durch Vue.js können dynamische Benutzeroberflächen erstellt werden, die reaktiv auf Benutzerinteraktionen reagieren, ohne dass die Seite neu geladen werden muss. Das bedeutet eine verbesserte Benutzererfahrung und ermöglicht die Implementierung von komplexen Funktionen wie Datenbindung, Ereignisbehandlung und Komponentenverhalten auf einfache Weise. Vue.js bietet außerdem nützliche Technologien wie Vuex [21] für die Zustandsverwaltung und den Vue Router [22] für das Routing in Single-Page-Anwendungen.

Bootstrap ist eine beliebte CSS- und JavaScript-Bibliothek, die eine Vielzahl von vorgefertigten Komponenten und Stilen bereitstellt. Dadurch können wir schnell und effizient ansprechende Benutzeroberflächen entwickeln, ohne jedes Detail von Grund auf neu erstellen zu müssen. Das Bootstrap-Grid-System [23] ist besonders hilfreich, um Layouts für verschiedene Bildschirmgrößen zu optimieren und die Anpassung an unterschiedliche Geräte und Bildschirmauflösungen zu erleichtern.

Darüber hinaus bietet Bootstrap eine umfangreiche Auswahl an Designvorlagen und Themes, die die Gestaltung eines konsistenten und professionellen Erscheinungsbilds für die Webanwendungen ermöglichen.

4.1 Startbildschirm

Abbildung 5: Mockup Webseite Startseite

Die Startseite der Plattform bietet Nutzern einen Überblick über sämtlich verfügbare Spiele. Durch die klare und gut strukturierte Navigationsleiste haben die Nutzer einfachen Zugang zu allen wichtigen Informationen. Hier findet man unter anderem die detaillierte Anleitung zur Nutzung der WebSockets, um eigene KI-Implementierungen mit dem Server zu verbinden, das Leaderboard, die Achievements (optional: bei Umsetzung des If-Time-Allows), sowie das Impressum der Plattform.

Oben rechts auf der Webseite befindet sich eine Suchfunktion, die selbst nach dem Hinzufügen neuer Spiele und zusätzlicher Funktionen eine schnelle und effiziente Navigation ermöglicht. Diese Suchfunktion wurde eingeführt, um die Benutzerfreundlichkeit zu steigern und die Plattform für mehr Spiele leichter erweiterbar zu machen.

Durch einen einfachen Klick auf eines der angebotenen Spiele gelangt der Nutzer direkt zur entsprechenden Spielelobby. Dort kann er sich weiter über das Spiel informieren und gegebenenfalls direkt damit interagieren.

4.2 Spielekonfiguration

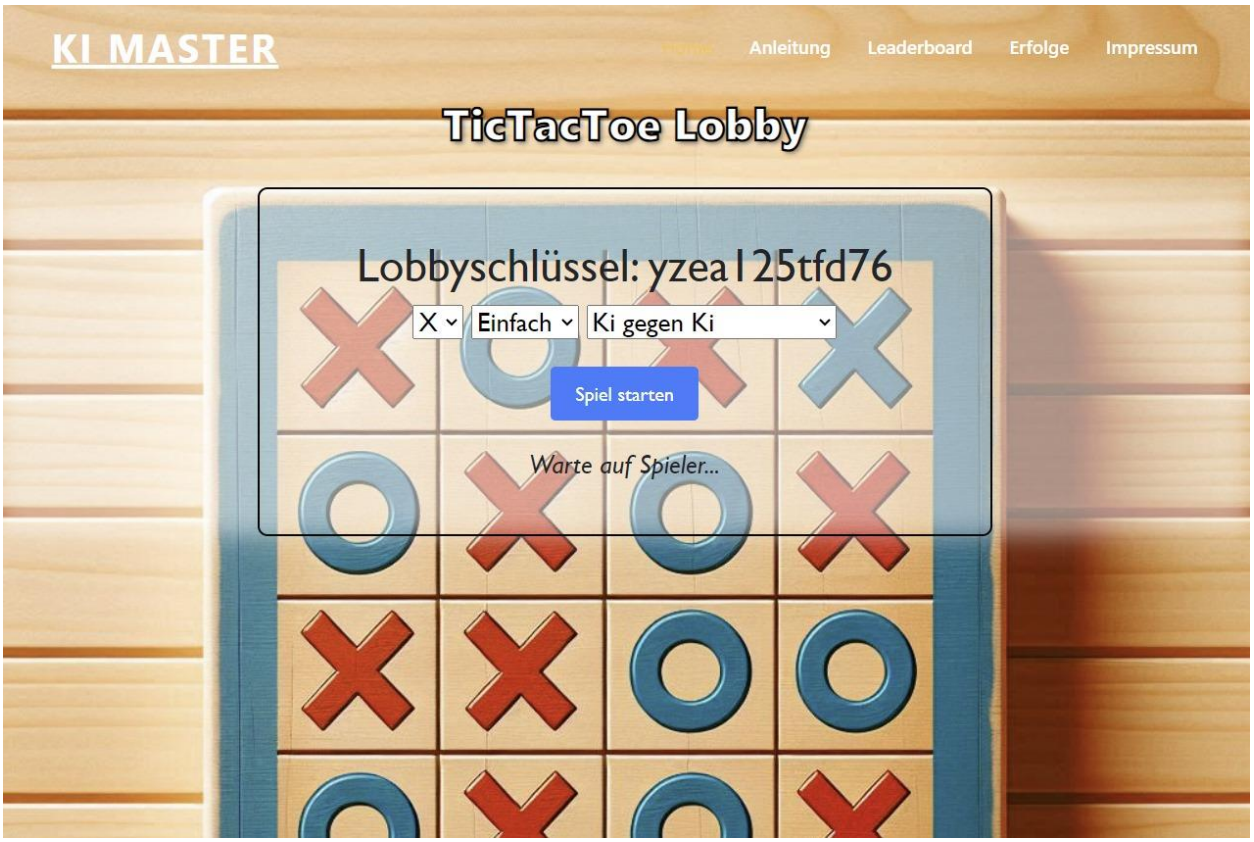

Abbildung 6: Mockup Webseite Spielelobby (hier: Tic-Tac-Toe)

In der Spielelobby hat der Benutzer die Möglichkeit, sein Spiel individuell zu konfigurieren. Dies beinhaltet die Auswahl der Steine oder Farben, die Einstellung des Schwierigkeitsgrads, wenn dieser für das jeweilige Spiel umgesetzt wurde und der Festlegung des Spielmodi, wie Spieler gegen Spieler, Spieler gegen Plattform-KI oder KI gegen KI.

Zusätzlich wird in der Lobby der Lobbyschlüssel angezeigt, der benötigt wird, um beispielsweise gegen andere Spieler anzutreten. Im Falle eines Mehrspielermodus wird angezeigt, ob bereits ein anderer Benutzer die Lobby betreten hat und das Spiel gestartet werden kann. Sind alle Spieler in der Lobby und bereit, wird durch „Spiel starten“ das eigentliche Spiel begonnen.

Die Navigationsleiste enthält eine Anleitung mit den Spielregeln des jeweiligen Spiels. Durch einen Klick auf das Logo in der linken oberen Ecke gelangt der Benutzer jederzeit zurück zur Startseite der Plattform.

4.3 Spieleoberfläche

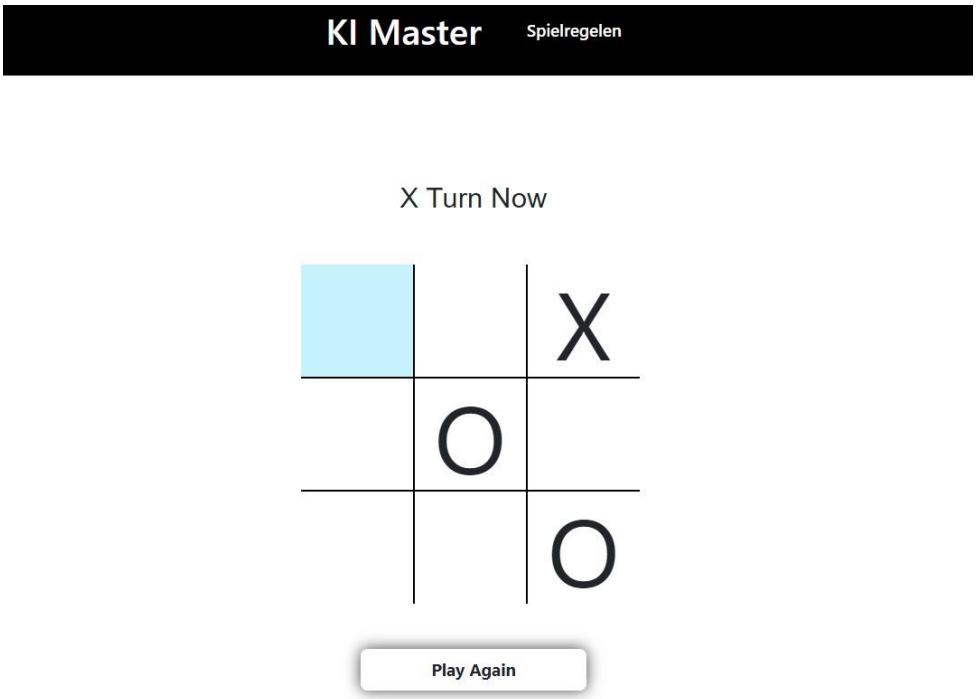

Abbildung 7: Mockup Webseite Spieleoberfläche (hier: Tic-Tac-Toe)

Die Spielseite präsentiert das Spielfeld des jeweiligen Spiels, das mit Text und einer dynamischen Benutzeroberfläche ausgestattet ist, um dem Spieler eine intuitive Navigation zu ermöglichen. Der Spieler kann beispielsweise durch das Markieren des ausgewählten Feldes oder durch präzise formulierte Anweisungen einfach durch das Spiel interagieren.

Je nach Spielmodus bietet die Spielseite dem Spieler die Möglichkeit, Spielzüge zurückzunehmen, was besonders nützlich ist, um strategische Entscheidungen zu überdenken und die Spielerfahrung zu verbessern, sowie Spiel aufgeben und neues Spiel starten.

Im Einzelspielermodus kann der Spieler das Spiel beliebig oft wiederholen, um sich zu verbessern oder neue Taktiken auszuprobieren, ohne Beschränkungen. Spielt er gegen einen anderen Benutzer, können auf gegenseitigen Konsens Spiele abgebrochen oder aufgegeben und neue Spiele begonnen werden. Verlässt ein Spieler im Zwei-Spielermodus den Raum und betritt ihn in den nächsten dreißig Sekunden nicht erneut, wird das Verlassen für den Gegner als Gewinn dargestellt und das Spiel beendet. Verlässt der zweite Spieler ebenfalls die Lobby, schließt sich der Raum automatisch.

In der Navigationsleiste sind äquivalent zur Lobby-Navigation Funktionen eingebettet.

5 Funktionale (Detail-) Anforderungen

5.1 Must Haves

5.1.1 Webseite

Über gängige Webbrowser erreichbare Webseite, die eine benutzerfreundliche Navigation bietet. Browserunterstützung findet für die aktuelleren Versionen von Firefox der Mozilla Foundation [24] und dem Google Browser Chrome [25] statt. Für andere Browser wird keine Funktionalität gewährleistet.

Sie sollte reaktionsschnell sein und Inhalte in Echtzeit laden (keine merkbaren Latenzen), um eine angenehme Erfahrung für Besucher sicherzustellen. Alle Funktionen und Inhalte der Webseite müssen klar strukturiert und leicht zugänglich sein, damit Benutzer problemlos durch die Seiten navigieren können. Wichtige Informationen sollten deutlich sichtbar platziert sein, um eine intuitive Nutzung zu ermöglichen.

Die Webseite muss ein gut sichtbares Impressum und klare Datenschutzbestimmungen enthalten, die den rechtlichen Anforderungen entsprechen. Diese Informationen sollten leicht auffindbar und verständlich formuliert sein, um Transparenz und Vertrauen bei den Nutzern zu fördern. Darüber hinaus müssen die Datenschutzbestimmungen die Erhebung, Verarbeitung und Nutzung von personenbezogenen Daten klar darlegen und die Rechte der Benutzer in Bezug auf ihre Daten deutlich beschreiben.

5.1.2 Externe Anwendung

Zusätzlich zur herkömmlichen Browser-Navigation soll es möglich sein, über eine Terminaleingabe über WebSockets mit dem dahinterliegenden Server zu interagieren. Diese Funktionalität ermöglicht es Benutzern, bestimmte Aktionen oder Abfragen direkt über eine programmatische Schnittstelle auszuführen, ohne die Webseite besuchen zu müssen. Die Kommunikation sollte klar dokumentiert und einfach zu nutzen sein, um eine reibungslose Interaktion mit dem System zu gewährleisten. Auf der Webseite wird zu diesem Zwecke die Verbindungsanleitung hinterlegt und bei Konfiguration über die Seite zu entsprechender Zeit präsentiert.

Debug Infos ?

5.1.3 Umgesetzte Spiele

Es werden verschieden komplexe Spiele angeboten, wobei eine Erweiterung des Spielepools durch Möglichkeiten der Skalierbarkeit im Backend bereitgestellt werden.

- ***Othello (Reversi)***: Strategisches Brettspiel für zwei Spieler. Ziel des Spiels ist es, die meisten Steine auf dem Spielbrett zu besitzen, indem man gegnerische Steine einkreist und sie in die eigene Farbe umdreht.
    - Die Basisimplementierung des alphazero frameworks [12], auf dessen Strukturen die Spiellogik entfernt basiert, erklärt seine Nutzung anhand des Spiels Othello. Die Implementierung der Spiellogik und das neurale Netz nutzen wir als Ansatz zum Trainieren einer eigenen KI.
- ***Tic-Tac-Toe***: Ist ein klassisches Zwei-Spieler-Spiel, bei dem die Spieler abwechselnd ihre Symbole (üblicherweise X und O) in einem Raster platzieren. Das Ziel ist es, drei Symbole in einer Reihe, Spalte oder Diagonale zu platzieren, bevor der Gegner dies schafft. Das Spiel endet entweder in einem Sieg, einer Niederlage oder einem Unentschieden.
    - Im alphazero frameworks [12] wird ebenfalls Tic-Tac-Toe umgesetzt, auf dessen Basis mit Änderungen an des darüberliegenden Ansprechens angeknüpft wird.
- ***Vier gewinnt***: Ist ein strategisches Brettspiel für zwei Spieler. Ziel des Spiels ist es, als Erster vier seiner eigenen Spielsteine horizontal, vertikal oder diagonal in einer Linie zu platzieren. Die Spieler wechseln sich ab, einen Spielstein in eine Spalte fallen zu lassen.
    - Im alphazero frameworks [12] wird ebenfalls Vier gewinnt umgesetzt, auf dessen Basis mit Änderungen an des darüberliegenden Ansprechens angeknüpft wird. Zudem müssen Änderungen des neuralen Netzes vorgenommen werden, aufgrund der Technologieänderungen von Keras.
- ***Nim***: Ist ein mathematisches Strategiespiel, bei dem zwei Spieler abwechselnd eine Anzahl von Objekten aus verschiedenen Haufen nehmen. Das Ziel ist es, den Gegner dazu zu bringen, den letzten Gegenstand zu nehmen.
    - Nim basiert auf keinen vorher bestehenden Projekten und wird vollständig autonom umgesetzt.
- ***Dame***: Ist ein klassisches Brettspiel für zwei Spieler. Die Spieler bewegen abwechselnd ihre Spielsteine diagonal über das Spielbrett. Ziel ist es, die gegnerischen Spielsteine zu schlagen, indem man über sie hüpft. Das Spiel endet, wenn ein Spieler alle gegnerischen Steine schlägt oder der Gegner nicht mehr ziehen kann.
    - Dame basiert auf keinen vorher bestehenden Projekten, orientiert sich aber an einer eigenen KI-Lösung basierend auf einem Monte-Carlo Tree Search.
- ***Go***: Ist ein strategisches Brettspiel für zwei Spieler. Die Spieler platzieren abwechselnd ihre Steine auf einem Gitterbrett, um Territorium zu beanspruchen und gegnerische Steine einzuschließen. Ziel ist es nach einem voll besetzten Brett oder einem Aufgeben, das meiste Territorium zu besitzen.
    - Mit Go als einer der Gründungsspiele von Alpha Zero wird es ebenfalls umgesetzt. Dabei wird auf eine Mischung zahlreicher vorhandener Implementierungen mit Adaption an die Spiellogik aus dem alphazero frameworks [12] zurückgegriffen.

- ***Waldmeister*** *[13]*: Ist ein taktisches Brettspiel, bei dem die Spieler um die größten Gruppen von gleichhohen oder gleichfarbigen Bäumen konkurrieren.
    - Waldmeister orientiert sich an einer fertig implementierten Lösung eines Mitstudierenden und eigenen Ansätzen.

5.1.4 Spielekonfiguration

Sowohl die Webseite als auch die Terminalunterstützung stellen eine Möglichkeit bereit neben dem zu spielenden Spiel weitere Einstellungen vorzunehmen.

So lassen sich zudem folgende Konfigurationen ändern:

- ***Spielmodus***: Zur Unterstützung vieler Anwendungsbereiche werden verschiedene Spielmodi angeboten. Dazu gehören, Spieler gegen Spieler, Spieler gegen von uns vortrainierte KIs, eine KI-Implementierung des Spielers gegen eine vorab trainierte KI, sowie zwei Spieler-KIs. Da die dahinterliegende Spiellogik und die Kommunikation über WebSockets generisch gestaltet sind, ist es für das Spiel nicht relevant, ob über die Webseitenanfragen, User-Anfragen oder die vortrainierten KIs gespielt wird. Lediglich die Repräsentation in das Frontend und damit die Weiterleitung der neuen Informationen differiert. Findet ein Spiel über die Webseite statt, so wird über den Websocket das Spielfeld auf der Webseite aktualisiert. Spielt jedoch eine Spieler-KI, so wird die Aktualisierung an diese mit dem aktuellen Brett und Zug über JSON vermittelt.
- ***Spielkonfigurationen***: Es wird eine Möglichkeit geben einige Einstellungen in Bezug auf das Spiel vorzunehmen. Dazu gehört vor allem das Auswählen der Spielsteine und damit der Wahl, welcher Spieler das Spiel beginnt.
- ***Schwierigkeitsgrad***: Zur Erweiterung der Spielemöglichkeiten werden verschieden gut trainierte KIs angeboten. Dies bedarf einer eigenen KI-Trainierung für jedes Spiel und jeden Schwierigkeitsgrad.

5.1.5 Spielebeitritt

Nach der Konfiguration eines Spiels, wird ein Raumtoken generiert, der es einem Mitspielenden ermöglicht dem Spieleraum beizutreten (im Spieler-Spieler Modus) oder seine KI-Implementierung über WebSockets mit diesem Raum zu verbinden (im Spieler-KI Modus). Dies ist auf zwei Spielende begrenzt.

5.1.6 Spielerepräsentation und -navigation

Da die Spiele sowohl über das Terminal als auch über die Webseite ausgeführt werden können, unterstützen beide Instanzen die Repräsentation des Spielfeldes.

In der Terminalnavigation werden Spielzüge mit entsprechenden Befehlen ausgeführt oder Verbindungsaktionen ausgeführt. Das Spielfeld wird dabei im Terminal dargestellt, sodass eine vollkommen von der Webseite unabhängige Spielerfahrung möglich ist.

Die Webseite unterstützt eine graphische Darstellung des Spiels. Dies wird durch PyGame, eine Python Bibliothek, gewährleistet. Zudem ermöglicht die Webseite eine mausgesteuerte Bedienung und Interaktion mit den Spielsteinen auf dem Spielfeld, sowie den zusätzlichen Aktionen der Seiten (bspw. Buttons, Suchleiste).

5.1.7 Unterstütze Spielefunktionen

Da einer Spieleinstanz spezifische Räume mit Spielen zugeordnet sind, ist es möglich, Spiele abzubrechen oder aufzugeben und dann mit einem neuen Spielanfang fortzufahren. Spieler haben dabei jederzeit die Option, einen Raum zu schließen und zur Hauptseite zurückzukehren, um entweder das aktuelle Spiel zu verlassen oder ein neues Spiel zu beginnen.

Zudem dienen zwei Optionen vor allem der genaueren Analyse von KI-Verhalten. So wird eine Möglichkeit der Mehrfachspiele für den KI-KI Modus geboten, bei der keine graphische Oberfläche unterstützt wird, jedoch eine statistische Auswertung der Siege und Niederlagen folgt. Zusätzlich lassen sich nach Vollendung eines Spiels die vergangenen Züge durch ein Zeitleistenverhalten betrachten.

5.2 Nice-To-Haves

5.2.1 Weitere Spiele

Es werden weitere verschieden komplexe Spiele zu dem Spielepool angeboten.

- ***Schach*:** Ist ein klassisches strategisches Brettspiel für zwei Spieler. Ziel ist es, den gegnerischen König matt zu setzen, was bedeutet, dass der König bedroht ist und keine Fluchtmöglichkeit hat. Die Spieler bewegen abwechselnd ihre Schachfiguren über das Spielbrett, wobei jede Figur spezielle Bewegungsmöglichkeiten hat.
- ***Mühle*:** Ist ein traditionelles Brettspiel für zwei Spieler. Das Ziel ist es, drei eigene Spielsteine in einer Reihe (Mühle) zu platzieren, wodurch man einen gegnerischen Stein entfernen kann. Hat ein Spieler keine Zugmöglichkeiten mehr, oder besitzt weniger als zwei Steine, hat er verloren.
- ***Halma*:** Ist ein strategisches Brettspiel für zwei bis sechs Spieler (in unserer Implementierung für zwei). Das Ziel ist es, alle eigenen Spielsteine in das gegenüberliegende Feld zu bringen, indem man über benachbarte Spielsteine springt.
- ***Abalone*:** Ist ein taktisches Brettspiel für zwei Spieler. Ziel ist es, sechs gegnerische Kugeln vom Spielbrett zu drängen, indem man mit eigenen Kugeln eine Linie bildet und die Kugeln des Gegners schiebt.

- ***Hex*:** Ist ein strategisches Zwei-Spieler-Brettspiel, bei dem das Ziel darin besteht, eine durchgehende Verbindung von einer Seite des Spielbretts zur gegenüberliegenden Seite zu schaffen, bevor der Gegner dies tut. Die Spieler platzieren abwechselnd ihre Steine auf einem sechseckigen Raster und versuchen, eine ununterbrochene Linie ihrer Farbe zu bilden.

5.2.3 Spielekonfiguration

- ***Spielfeldoptionen***: Bei einfacheren Spielen wie Tic-Tac-Toe, Othello und Vier gewinnt werden unterschiedlich große Spielfelder angeboten, sodass nicht nur die Standardgrößen, wie 3x3 oder 4x4 angeboten werden. Dies bedarf einer Spielfeldkonfiguration in der Spiellogik der jeweiligen Spiele und einer eigenen KI-Trainierung.

5.2.3 Webseitenerweiterung

Zur Unterstützung des Spielspaß und Belohnung guter Spiele und KI-Implementierungen wird eine Anzeigetafel mit Punktedarstellungen der in einem Spiel erspielten Punkte von KI oder Spieler hinzugefügt. Um dies zu realisieren werden Benutzeridentifikationen mit der Rückmeldung eines Benutzernamens eingeführt.

Es wird eine Möglichkeit bereitgestellt, Spiele für spätere Analysen und Fortsetzungen des Spiels abzuspeichern. Dazu gehört zudem eine Lademöglichkeit der abgespeicherten Spiele. Dazu wird keine Benutzeridentifikation benötigt.

Es wird ein Zuschauermodus unterstützt, der abhängig von Konfiguration und Beitrittsreihenfolge entschieden wird. Dies ermöglicht die Observierung von KI-Spielen mit einer graphischen Oberfläche.

Die Webseite steht zudem in einem weiteren Farbmodus zur Verfügung, sodass nicht nur ein Light-Mode, sondern auch ein Dark-Mode angeboten werden. Die Einstellung lassen sich auf den verschiedenen Seiten der Webseite intuitiv anwählen.

5.3 If-Time-Allows

5.3.1 Zufallsspiele

Es werden weitere Spiele zu dem Spielpool hinzugefügt. Diese beinhaltet jedoch den alphazero Formalien nach nicht regelkonformen Bedingungen. Sie werden bestimmt durch Zufallskomponenten im Spiel (bspw. Würfel).

5.3.2 Webseitenerweiterung

Zur Erweiterung der Webseite, wird zunehmend die Barrierefreiheit der Webseite ausgebaut. Sie unterstützt dabei nicht nur Screen Reader, sondern auch vollständig mausfreie Bedienung. Dazu zählt auch die Bereitstellung der Webseite für mobile Geräte, die die Spieleoberfläche und Navigation in einer Toucheingabe erweitern, und eine Mehrsprachenunterstützung der Webseitentexte und Anleitungen in mindestens Englisch und Deutsch.

Zusätzlich wird ein Achievement-System mit speziellen Zielen, Meilensteinen, Belohnungen und Rängen eingeführt. Um dieses System zu unterstützen, wird die Benutzeranmeldung erweitert, um die Speicherung der erhaltenen Ränge und Erfolge zu ermöglichen. Jeder Benutzer wird ein Profil haben, das seine errungenen Achievements und Fortschritte verfolgt. Dieses Profil bietet dem Benutzer eine Übersicht über seine Erfolge und schafft Anreize für eine kontinuierliche Nutzung der Plattform.

6 Qualitative Anforderungen

6.1 Allgemeine Anforderungen

Für die Benutzeroberfläche wird eine hohe Benutzerfreundlichkeit angestrebt, wodurch eine intuitive Navigation und Interaktion gewährleistet werden sollen. Dies umfasst klare und verständliche Menüstrukturen, Schaltflächen und Layouts, die den Nutzern eine einfache und angenehme Bedienung ermöglichen. Zudem sollen Fehlermeldungen präzise und informativ sein, um den Benutzern bei auftretenden Problemen eine effiziente Fehlerbehebung zu ermöglichen. Das System soll eine hohe Zuverlässigkeit aufweisen und eine hohe Verfügbarkeit sicherstellen, um Unterbrechungen im Betrieb auf ein Minimum zu reduzieren.

Zu dessen Sicherstellung sollen Usability-Tests bei Zielgruppen durchgeführt werden, sodass auf dessen Feedback in folgendem eingegangen werden kann.

6.2 Gesetzliche Anforderungen

Das System muss sämtlichen relevanten gesetzlichen Vorgaben und Bestimmungen entsprechen, insbesondere in Bezug auf Datenschutz und Sicherheit. Dies beinhaltet die Einhaltung der DSGVO [14] (Datenschutz-Grundverordnung) sowie anderer lokaler Datenschutzgesetze und -vorschriften.

Kritische Faktoren sind in diesem Zusammenhang die Verbindung der Benutzer über die Webseite und über die externe Anwendung. Zudem die Zwischenspeicherung von Spielen für die Zugrücknahme, die mögliche Speicher- und Ladefunktion von Spielen und die Eingabe eines Benutzernamens auf der Webseite.

Zur Kommunikation über WebSockets und Identifikation der Spieler und bespielten Räume sind Tokens nötig. Ihre Speicherung ist nur für die interne Verwendung kritisch und auch nur relevant, solange die Docker-Container der Räume existieren (also Session gebunden). Zudem lassen sich daraus keine benutzerspezifischen Informationen ziehen.

Da die Zwischenspeicherung von Spielen zur Zugrücknahme ebenfalls an den Token geknüpft ist, sind dessen Daten in diesem Zusammenhang ebenfalls unkritisch und nicht benutzerspezifisch.

Die Funktion des Speicherns und Ladens von Spielen (Nice-To-Have) muss dem Benutzer die Möglichkeit geben, nur bei konkreter Einwilligung und mit verfügbarer Aufklärung über gespeicherte Daten und den Zeitraum der Speicherung ein Speichern zu bewilligen oder zu verbieten.

Für die Ranglisten (Nice-To-Have) besteht ja die Möglichkeit Benutzernamen anzugeben, die dann mit Tokens in Verbindung gebracht werden können. Hier muss ebenfalls klar definiert sein, in welcher Form Daten gespeichert und weiterverarbeitet werden und eine Möglichkeit geboten werden, keinen Benutzernamen anzugeben.

Die Benutzerverwaltung mit Authentifizierung und Login Möglichkeiten für die Achievements (If-Time-Allows) bedürfen einer sorgfältigeren Betrachtung. Da kritische personenbezogene Daten (beispielsweise eine E-Mail-Adresse und Passwortzuordnung) verarbeitet und gespeichert werden. Für den korrekten Umgang muss das System angemessene Sicherheitskriterien erfüllen, um die Daten zu schützen, was wiederum einen Mehraufwand in Bezug auf Sicherheit für die Plattform darstellt. Zudem dürfen diese nur für bestimmte Verarbeitungszwecke und -zeiträume mit Möglichkeiten zum Widerruf und Möglichkeit der Einsicht der gespeicherten Daten und Verarbeitungsketten verarbeitet werden und man benötigt Mechanismen zur konkreten Einwilligung der betroffenen Personen.

6.3 Technische Anforderungen

Das System soll plattformübergreifend funktionsfähig sein und auf einer Vielzahl von nicht mobilen Endgeräten (If-Time-Allows umfasst auch mobile Nutzung) nahtlos und fehlerfrei laufen. Dazu findet Browserunterstützung für die aktuelleren Versionen von Firefox der Mozilla Foundation und den Google Browser Chrome statt. Für andere Browser wird keine Funktionalität gewährleistet.

6.4 Weitere Anforderungen

Das System soll skalierbar sein, um zukünftiges Wachstum und eine steigende Anzahl von Benutzern zu unterstützen, ohne die Leistung zu beeinträchtigen. Dazu werden die Docker-Container und Tokens der Räume verwendet. Sie ermöglichen erst einmal eine unbegrenzte Menge an gleichzeitigen Spielen und Nutzern (mit einer ersten Beschränkung von 65525 aktiven

Benutzern. Einschränkungen finden statt durch die Verbindung, die durch Netzwerkverfügbarkeit und Serverauslastung beruhen.

Zusätzlich soll die Client-Server Architektur mit kombinierten Microservices die Flexibilität für zukünftige Erweiterungen und Anpassungen bereitstellen. Wobei neue Spiele und KI-Lösungen umgesetzt werden können. Dieser Prozess soll zudem durch die konkrete (Weiterentwicklungs-) Dokumentation unterstützt werden.

7 Umfang der Anforderungen

Komponente	Zeitabschätzung	Entwickler
Must-Haves		
- Webseite		
▪ Webseitendesign	35 Stunden	Omar Karkotli, Pascal Waldschmidt
▪ Kommunikation vom und zum FastAPI-Server	15 Stunden	Pascal Waldschmidt, Alexander Roos
▪ Verarbeitung der Benutzereingaben und -interaktionen	10 Stunden	Omar Karkotli, Pascal Waldschmidt
▪ Impressum und Datenschutzbestimmungen rechtlich korrekt formulieren und auf der Webseite darstellen	10 Stunden	Omar Karkotli
▪ Verbindungsdokumentation hinterlegen und anzeigen	2 Stunden	Omar Karkotli
- Externe Anwendung		
▪ Kommunikationsschnittstelle aufsetzen (in Zusammenhang mit Webseitenkommunikation)	25 Stunden	Maximilian Bachmann, Alexander Roos
▪ Verbindungsmöglichkeiten über WebSockets bereitstellen (Befehle & Codeausschnitte)	10 Stunden	Maximilian Bachmann, Alexander Roos
▪ Dokumentation der Verbindungen für Benutzer	15 Stunden	Maximilian Bachmann, Justine Buß
- Umgesetzte Spiele		
▪ Spiellogik implementieren durch Umsetzen der Spielregeln (Feldgröße, Gewonnen/Verloren/	Je nach Spiel zwischen 4 bis 12	Justine Buß, Alexander Roos, Maximilian

Unentschieden, Spiegelungen, Symmetrien, Spielablauf, ...)	Stunden pro Spiel	Bachmann, Pascal Waldschmidt
▪ Logik des neuralen Netzes implementieren (Keras, TensorFlow, PyTorch)	Je nach Spiel zwischen 8 bis 10 Stunden pro Spiel	Justine Buß, Alexander Roos, Maximilian Bachmann, Pascal Waldschmidt
▪ Trainieren der KIs (auch in verschiedenen Schwierigkeits-stufen) mit Abspeicherung der Daten in einem geeigneten Format	Je nach Spiel etwa 1 Tag pro Spiel Trainieren	Alexander Roos
- Spielekonfiguration		
▪ Spielmodus: Generische Implementierung des Spiels für Spielerauswahl	6 Stunden	Maximilian Bachmann
▪ Spielkonfiguration: Dynamische Repräsentation für bspw. Spielsteinwahl	5 Stunden pro Spiel + etwa 1 Tag pro Spiel Trainieren	Pascal Waldschmidt, Maximilian Bachmann
▪ Schwierigkeitsgrad: Trainieren verschieden konfigurierter Versionen von KIs (Wahrscheinlichkeitsvektor zur Spielzugwahl)	3 Stunden	Alexander Roos
▪ Darstellung im Frontend (Räume und Konfigurationen)	10 Stunden	Omar Karkotli, Pascal Waldschmidt
- Spielebeitritt		
▪ Schlüsselgenerierung der Räume und Benutzer	2 Stunden	Alexander Roos
▪ Lobby Bereitstellung	10 Stunden	Alexander Roos, Pascal Waldschmidt
▪ Weiterleitung an Clients	3 Stunden	Maximilian Bachmann
- Spielerepräsentation und -navigation		
▪ Terminalrepräsentation von Spielen	3 Stunden pro Spiel	Justine Buß, Maximilian Bachmann
▪ Spielbrett in PyGame schön darstellen	6 Stunden pro Spiel	Justine Buß, Maximilian Bachmann
▪ Bereitstellungen von Benutzerinteraktionsfeldern	10 Stunden	Omar Karkotli, Pascal Waldschmidt
- Unterstütze Spielefunktionen		

▪ Implementierung in der Spiellogik von Abbruch, Aufgeben, neues Spiel beginnen, Lobby schließen	5 Stunden	Justine Buß, Maximilian Bachmann
▪ Mehrfachspieloption mit statistischer Auswertung	10 Stunden	Justine Buß, Alexander Roos, Maximilian Bachmann
▪ Zeitleistenverhalten mit Möglichkeit X Züge rückgängig zu machen	15 Stunden	Justine Buß, Alexander Roos, Maximilian Bachmann
Nice-To-Haves		
- Weitere Spiele	Je nach Spiel etwa 10 bis 15 Stunden Spiellogik + 10 Stunden für NNet + 1 Tag Trainieren pro Spiel	Justine Buß, Alexander Roos, Maximilian Bachmann, Pascal Waldschmidt
- Spielekonfiguration		
▪ Spielfeldoptionen: Spiellogik in Variationen anbieten und trainieren (Dynamische Spielfeldgrößen 3x3, 4x4,...)	6 Stunden pro Spiel + etwa 1 Tag pro Spiel Trainieren	Justine Buß, Alexander Roos, Maximilian Bachman
- Webseitenerweiterung		
▪ Anzeigetafel mit Punktedarstellung auf der Webseite	2 Stunden	Omar Karkotli
▪ Benutzeridentifikationen im Frontend	4 Stunden	Omar Karkotli, Pascal Waldschmidt
▪ Benutzeridentifikationsverarbeitung im Backend	9 Stunden	Thorben Jones
▪ Spielspeicherungsoption im Frontend	4 Stunden	Omar Karkotli, Pascal Waldschmidt
▪ Spielspeicherungsumsetzung im Backend	6 Stunden	Sven Reinhard
▪ Zuschauermodus in den Lobbys	7 Stunden	Alexander Roos, Pascal Waldschmidt
▪ Farbmoduswahl (Light-Mode, Dark-Mode)	6 Stunden	Omar Karkotli, Maximilian Bachmann
If-Time-Allows		

- Zufallsspiele	40 Stunden + 15 Stunden Spiellogik pro Spiel	
- Webseitenerweiterung		
▪ Barrierefreiheit (WCAG 2.1 AA und mausfreie Bedienung)	15 bis 20 Stunden	Sven Reinhard, Omar Karkotli
▪ Bereitstellung für mobile Geräte	30 Stunden	Omar Karkotli, Pascal Waldschmidt, Sven Reinhard
▪ Mehrsprachenunterstützung	10 Stunden	Justine Buß, Omar Karkotli, Sven Reinhard
- Achievement-System		
▪ Benutzeranmeldung ausbauen und Login ermöglichen mit Passwortverwaltung/-speicherung und Sicherheitsaspekten	50 Stunden	Justine Buß, Alexander Roos, Maximilian Bachmann, Pascal Waldschmidt
▪ Umsetzung auf der Webseite	10 Stunden	Omar Karkotli
Sonstiges		
- Dockerumgebung		
▪ Dockerfiles formulieren	4 Stunden	Sven Reinhard, Thorben Jones
▪ Dockernetzwerk aufsetzen und verwalten	25 Stunden	Sven Reinhard, Thorben Jones
▪ Docker-compose.yml schreiben	3 Stunden	Sven Reinhard, Thorben Jones
▪ Kommunikation zwischen Containern bereitstellen	8 Stunden	Sven Reinhard, Thorben Jones
- Tests		
▪ Testfallerstellung	15 Stunden	Sven Reinhard, Thorben Jones
▪ Testautomatisierung und manuelle Tests	25 Stunden	Sven Reinhard, Thorben Jones
▪ Usability Tests	15 Stunden	Thorben Jones
- Dokumentation		
▪ Benutzerhandbuch	8 bis 10 Stunden	Justine Buß, Omar Karkotli, Thorben Jones

▪ Entwicklerdokumentation	20 bis 25 Stunden	Maximilian Bachmann, Alexander Roos
▪ Weiterentwicklungsdokumentation	10 bis 12 Stunden	Justine Buß, Maximilian Bachmann, Alexander Roos
▪ Testdokumentation	20 Stunden	Sven Reinhard, Thorben Jones

Tabelle 2: Umfang der Anforderung

Summe Must have + Pos
Tests + Sonstiges ?

8 Rahmenbedingungen

8.1 Zeitplan

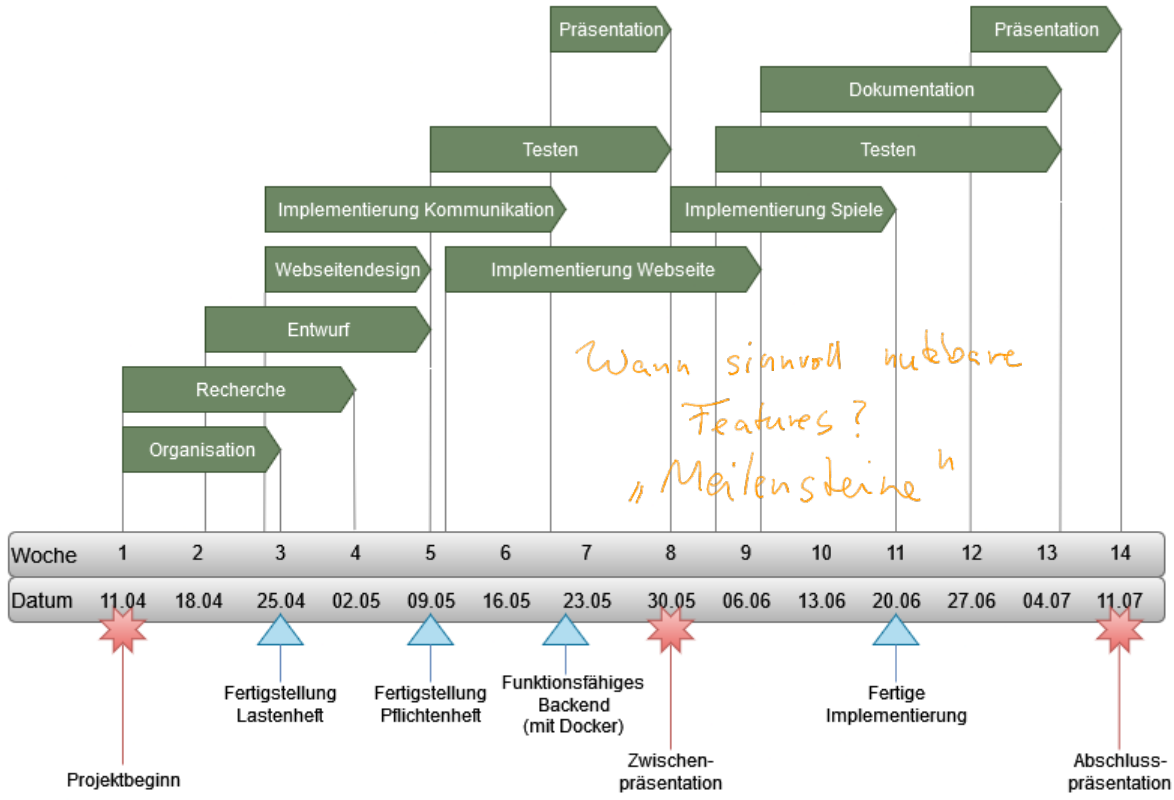

Abbildung 8: Meilensteindiagramm Projektplanung

Das Projekt erstreckt sich über einen Zeitraum von drei Monaten, von Mitte April bis Mitte Juli. Der offizielle Projektstart ist am 11. April mit einem Kickoff-Meeting und endet mit einer abschließenden Projektpräsentation und Abnahmen.

In den ersten Wochen liegt der Fokus auf der Einarbeitung in das Projekt und den Entwurf der grundlegenden Projektstruktur. Dabei stellen das Lastenheft und Pflichtenheft die bedeutenden Meilensteine der Entwurfsphase dar. Zur Umsetzung beider Dokumente werden erste technische Entwürfe und Implementierungen mit Featurebesprechungen und Recherchearbeiten verbunden.

Sobald der Entwurf abgeschlossen ist, erfolgt die Finalisierung der bereits begonnenen Kommunikationsschnittstellen des Projekts in Bezug auf alle Schnittstellen, Kommunikationsprotokolle und Docker-Netzwerke. Die Implementierung der Webseitendesign-Entwürfe sowie die Trainierung und Lauffähigmachung der ersten Spiele finden statt, um einen ersten Prototyp des Projekts für die Zwischenpräsentation Ende Mai zu testen und präsentabel zu machen.

Von der Rückmeldung der Präsentation an, liegt der Fokus auf der Implementierung. Die restlichen Spiele werden nacheinander umgesetzt, die Webseite fertiggestellt und systematische Tests begonnen. Bis Mitte und Ende Juni soll die Implementierungsphase abgeschlossen sein und alle Must-Have Features implementiert sein. Währenddessen stehen bereits intensive Tests und umfangreiche Dokumentation der Software an.

In den letzten Wochen vor der Abschlusspräsentation am 11. Juli liegt der Fokus auf der Fertigstellung der Website, abschließenden Tests und der Vorbereitung der endgültigen Präsentation. Diese beendet das Projekt für unser Team mit einer Projektabnahme und -vorstellung.

8.2 Technische Anforderungen

9.2.1 Server Spezifikationen

Der Server muss mindestens mit einer Quad-Core CPU ausgestattet sein, um die erforderliche Rechenleistung für die Netzwerkbearbeitung und die Ausführung der Spielinstanzen zu gewährleisten. Die aktuelle Architektur des internen Python-Netzwerks erlaubt theoretisch die gleichzeitige Bearbeitung von bis zu 65525 Benutzern, was eine hohe Mehrspielermöglichkeit bereitstellt. Jede einzelne Spielinstanz wird voraussichtlich eine Größe von circa 2 MB haben.

(Woher die Zahl?)

9.2.2 Software-Anforderungen

Die erforderlichen Bibliotheken sind in den `requirements.txt` Dateien des Haupt- und Netzwerk-Branches aufgeführt. Diese müssen für die korrekte Ausführung und Kommunikation des Systems installiert sein. Die wichtigsten sind:

- fastapi==0.111.0
- fastapi-cli==0.0.2

- httpcore==1.0.5
- httptools==0.6.1
- httpx==0.27.0
- numpy==1.26.4
- pygame==2.5.2
- ujson==5.9.0
- websockets==12.0

9.2.3 Netzwerk- und Sicherheitsanforderungen

Zur Sicherung der Kommunikation und des Netzwerks, sind alle Netzwerkaktivitäten innerhalb des Netzwerks aus Docker-Containern nicht nach außen geroutet und so für externe Nutzer nicht sichtbar. Die Kommunikation unter den Docker-Containern ist somit intern.

Der Server und die Schnittstelle nach außen zu den Clients akzeptiert zudem nur vordefinierte Befehle von autorisierten und Quellen. Unbekannte Befehle und Nachrichten werden abgelehnt oder mit einer entsprechenden Meldung ("Unknown") beantwortet.

9.2.4 Leistungsanforderungen

Nach einer initialen Verbindungszeit von maximal 3 Sekunden für die Game Clients sollte die Verzögerung während der Nutzung nahezu unmerklich sein. Bisher wurden jedoch keine umfassenden Performance-Tests außerhalb eines lokalen Netzwerks durchgeführt.

8.3 Problemanalyse

8.3.1 Integration und Skalierung von Komponenten

Die nahtlose Integration und Skalierung der Systemkomponenten erfordern eine präzise Verwaltung und Koordination durch den Server. Insbesondere müssen die Interaktionen zwischen den Komponenten (Webserver, Docker-Container, externe Dienste) effizient verwaltet werden, um Engpässe und Leistungsprobleme zu vermeiden.

Detaillierte Probleme:

- Kommunikationsengpässe zwischen verschiedenen Diensten
- Skalierung von Docker-Containern unter Berücksichtigung von Ressourcenverbrauch und Lastspitzen
- Effiziente Verteilung von Arbeitslasten auf mehrere Docker-Container
- Koordination der Interaktionen zwischen verschiedenen Mikroservices

Herangehensweise:

- Durch die Verwendung von Docker-Netzwerken können Dienste in isolierten Umgebungen ausgeführt werden, wodurch eine klare Trennung und sichere Kommunikation zwischen den Diensten gewährleistet wird. Dies erleichtert die Integration und Skalierung der Komponenten, da sie unabhängig voneinander betrieben werden können.

8.3.2 Dynamische und skalierbare WebSocket-Kommunikation

Die Implementierung einer robusten Websocket-Kommunikation erfordert die Bewältigung von Echtzeitnachrichten und die Skalierung der Kommunikation auf eine große Anzahl von gleichzeitigen Verbindungen.

Detaillierte Probleme:

- Management von Echtzeitnachrichten und -aktualisierungen zwischen Server und Client
- Behandlung von Verbindungsabbrüchen und Wiederaufnahmen
- Skalierung der Websocket-Infrastruktur, um eine große Anzahl von gleichzeitigen Benutzern zu unterstützen
- Sicherstellung der Stabilität und Leistung der WebSocket-Verbindungen

Herangehensweise:

- FastAPI ist ein Framework, das die Integration von WebSocket-Endpunkten und -Instanzen unterstützt. Dadurch können Echtzeitnachrichten zwischen Server und Clients asynchron senden und empfangen, was eine effiziente Websocket-Kommunikation ermöglicht.
- Die Verwendung von JSON für die Übertragung von Daten über WebSockets erleichtert die Verarbeitung und Interpretation der Nachrichten. Dies trägt dazu bei, die Leistung zu optimieren und die Kommunikation zwischen Server und Clients zu verbessern.

8.4 Qualitätssicherung

8.4.1 Unit-Tests

Für jede Komponente des Systems werden Unit-Tests implementiert, um deren isolierte Funktionalität zu überprüfen. Dabei werden spezifische Testfälle entwickelt, um sicherzustellen, dass die einzelnen Module gemäß den Anforderungen arbeiten und korrekt miteinander interagieren. Die Unit-Tests werden automatisiert ausgeführt, um eine effiziente und wiederholbare Überprüfung sicherzustellen.

8.4.2 Integrationstests

Die Integrationstests fokussieren sich auf die Interaktion und Kommunikation zwischen den Systemkomponenten. Insbesondere werden die Kommunikation über WebSockets zwischen dem Server und den Client-Anwendungen sowie die Verbindung zwischen dem Server und den

Docker-Containern getestet. Ziel ist es, sicherzustellen, dass alle Schnittstellen und Interaktionen reibungslos funktionieren und alle Systemkomponenten ordnungsgemäß zusammenarbeiten. Zur Unterstützung wird die Postman Software [16] ermöglicht eine abdeckende Endpunkttestung mit Funktionen für das Senden verschiedener Arten von HTTP-Anfragen

8.4.3 Sicherheitstests

Die Sicherheit des Systems wird durch gezielte Sicherheitstests überprüft. Dabei werden potenzielle Schwachstellen und Sicherheitslücken in der Kommunikation über WebSockets, der Verwaltung von Docker-Containern und anderen kritischen Bereichen identifiziert und behoben. Ziel ist es, die Vertraulichkeit, Integrität und Verfügbarkeit des Systems zu gewährleisten.

8.5 Dokumentation

Die umfassende Dokumentation am Ende des Projekts ist von entscheidender Bedeutung, um den reibungslosen Betrieb der Plattform sicherzustellen und zukünftige Entwicklungen zu unterstützen. Sie dient somit als wichtige Referenz für Benutzer, Entwickler und Testteams und unterstützt eine effiziente Nutzung, Wartung und Weiterentwicklung der Plattform im gesamten Lebenszyklus. Sie umfasst verschiedene Komponenten:

- ***Benutzerhandbuch***: Das Benutzerhandbuch enthält detaillierte Anleitungen zur Nutzung der Plattform und ihrer Funktionen. Es bietet den Benutzern klare und verständliche Informationen darüber, wie sie sich anmelden, Spiele spielen, Einstellungen ändern und andere interaktive Funktionen der Plattform nutzen können. Das Ziel ist es, den Benutzern eine einfache und effiziente Erfahrung zu ermöglichen.
- ***Entwicklerdokumentation***: Die Entwicklerdokumentation enthält wichtige Informationen zur Plattformarchitektur, WebSocket-Spezifikationen und Implementierungsdetails. Sie richtet sich an Entwickler und technisches Personal, die mit der Plattform arbeiten oder sie weiterentwickeln möchten. Diese Dokumentation erleichtert die Wartung, Erweiterung und Integration neuer Funktionen in die Plattform.
- ***Weiterentwicklungsdokumentation***: Die Weiterentwicklungsdokumentation bietet Anleitungen und Richtlinien zur Erweiterung und Weiterentwicklung der Plattform. Dies umfasst beispielsweise die Integration neuer Spiele, die Verbesserung der Webseitenzugänglichkeit oder die Implementierung zusätzlicher Funktionen. Die Dokumentation soll Entwicklern klare Schritte und Best Practices bieten, um die Plattform erfolgreich zu erweitern und anzupassen.
- ***Testdokumentation***: Die Testdokumentation enthält Berichte über durchgeführte Tests und deren Ergebnisse. Sie dokumentiert den gesamten Testprozess, einschließlich der Teststrategie, Testfälle, durchgeführten Testszenarien und der daraus resultierenden Testergebnisse. Diese Dokumentation ist entscheidend, um die Qualität und Zuverlässigkeit

der Plattform zu gewährleisten und etwaige Probleme oder Schwachstellen frühzeitig zu erkennen und zu beheben.

9 Abkürzungsverzeichnis

Abkürzung	Beschreibung
CSS	Cascading Style Sheets
JSON	JavaScript Object Notation
~~KI~~/AI	Künstliche Intelligenz/Artificial Intelligence
SHA	Secure Hash Algorithm
SI-Container	Spielinstanz-Container
TCP	Transmission Control Protocol

Tabelle 3: Abkürzungsverzeichnis

10 Anhang

1. Technische Hochschule Mittelhessen (THM), Fachbereich MNI (Medieninformatik und Medientechnik). Abgerufen von https://www.thm.de/mni/ [Zuletzt aufgerufen: 30. April 2024].
2. Kammer, Frank. Prof. Dr. Frank Kammer – Technische Hochschule Mittelhessen. Abgerufen von https://www.thm.de/mni/frank-kammer [Zuletzt aufgerufen: 30. April 2024].
3. Python Software Foundation. (2024). Python. Abgerufen von https://www.python.org/ [Zuletzt aufgerufen: 30. April 2024].
4. Docker Inc. (2024). Docker. Abgerufen von https://www.docker.com/ [Zuletzt aufgerufen: 30. April 2024].
5. Martín Abadi, et al. (2022). Abgerufen von https://www.tensorflow.org/ [Zuletzt aufgerufen: 30. April 2024].
6. Chollet, F., et al. Keras. Abgerufen von https://keras.io/ [Zuletzt aufgerufen: 30. April 2024].
7. David Silver, et al. (2018). AlphaZero: Shedding new light on chess, shogi, and Go. Abgerufen von https://deepmind.google/discover/blog/alphazero-shedding-new-light-on-chess-shogi-and-go/ [Zuletzt aufgerufen: 30. April 2024].
8. Tiangolo. FastAPI. Abgerufen von https://fastapi.tiangolo.com/ am 30. April 2024.
9. Pygame. News – Pygame v2.1.2 documentation. Abgerufen von https://www.pygame.org/news [Zuletzt aufgerufen: 30. April 2024].
10. Socket.IO. (2024). Abgerufen von https://socket.io/ [Zuletzt aufgerufen: 30. April 2024].
11. Silver, D., et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Abgerufen von https://discovery.ucl.ac.uk/id/eprint/10069050/1/alphazero_preprint.pdf [Zuletzt aufgerufen: 30. April 2024].
12. Nair, S, et al. (2024). suragnair/alpha-zero-general. Abgerufen von https://github.com/suragnair/alpha-zero-general [Zuletzt aufgerufen: 30. April 2024].
13. Gerhards Spiel und Design. (n.d.). Waldmeister. Abgerufen von https://www.spielewerkstatt.eu/de/strategie-taktik/204-waldmeister.html [Zuletzt aufgerufen: 30. April 2024].
14. DSGVO-Gesetz. (2018). DSVGO Datenschutz-Grundverordnung. Abgerufen von https://dsgvo-gesetz.de/ [Zuletzt aufgerufen: 30. April 2024].
15. PyTorch. (n.d.). Abgerufen von https://pytorch.org/ [Zuletzt aufgerufen: 30. April 2024].
16. Postman. (2024) Abgerufen von: https://www.postman.com/. [Zuletzt aufgerufen: 02.05.2024]
17. You, E., et al. (2024) Vue.js. Abgerufen von: https://vuejs.org/. [Zuletzt aufgerufen: 07.05.2024]

18. Otto, M., et al. (2024) Bootstrap v5.3.3. Abgerufen von: https://getbootstrap.com/. [Zuletzt aufgerufen: 07.05.2024]
19. Internet Engineering Task Force (IETF). (2011). The WebSocket Protocol. RFC 6455. Abgerufen von: https://datatracker.ietf.org/doc/html/rfc6455. [Zuletzt aufgerufen: 07.05.2024]
20. Docker Inc. (2023) Docker SDK for Python. Abgerufen von: https://docker-py.readthedocs.io/en/stable/. [Zuletzt aufgerufen: 07.05.2024]
21. You, E., et al. (2024) What is Vuex? Abgerufen von: https://vuex.vuejs.org/. [Zuletzt aufgerufen: 07.05.2024]
22. You, E., et al. (2024) Vue Router. The official Router for Vue.js. Abgerufen von: https://router.vuejs.org/. [Zuletzt aufgerufen: 07.05.2024]
23. Otto, M., et al. (2024) Grid system. Abgerufen von: https://getbootstrap.com/docs/4.0/about/overview/. [Zuletzt aufgerufen: 07.05.2024]
24. Mozilla Foundation. (2024). Firefox Browser: schneller, sicherer, smarter. Abgerufen von: https://www.mozilla.org/de/firefox/new/. [Zuletzt aufgerufen: 07.05.2024]
25. Google. Google Chrome herunterladen. Abgerufen von: https://www.google.com/chrome/de/download-chrome/?brand=FKPE&ds_kid=43700074554984216&gad_source=1&gclsrc=ds. [Zuletzt aufgerufen: 07.05.2024]