Projektskizze

Entwicklung eines Bundesliga-ChatBots

Vorgelegt bei
Prof. Dr. Thomas Farrenkopf

Technische Hochschule Mittelhessen
Fachbereich MND

16. Juni 2025
Sommersemester 2025

Kamal Badawi	5236028
Abdal Ahmad	5566419

Beschreibung

Das Projekt „Bundesliga-ChatBot“ beschreibt die Entwicklung einer Webanwendung, die aktuelle Spieldaten der 1. Bundesliga bereitstellt und interaktive Funktionen wie einen Chatbot bietet. Die Anwendung nutzt eine Client-Server-Architektur mit einem FastAPI-Backend, einem React-Frontend und einer MongoDB-Datenbank. Daten werden über eine externe API (OpenLigaDB) abgerufen und durch einen ETL-Prozess (Extract, Transform, Load) in ein für das Large Language Model (LLM) Gemini 1.5 Flash nutzbares Format transformiert. Zur effizienten und skalierbaren Datenverarbeitung wird Retrieval-Augmented Generation (RAG) eingesetzt. Bei RAG werden strukturierte und unstrukturierte Bundesligadaten in Vektoren umgewandelt und in einer spezialisierten Vektordatenbank gespeichert, die semantische Nähe repräsentiert. Nutzerfragen werden ebenfalls in Vektoren umgewandelt, und eine semantische Suche identifiziert die relevantesten Inhalte, die dem LLM zur Generierung kontextbezogener Antworten bereitgestellt werden. Dies reduziert die Token-Anzahl, senkt Kosten und verbessert die Antwortzeiten. Benutzer können Fragen zu Spieltagen, Ergebnissen und Tabellen stellen, die vom LLM basierend auf den RAG-retrieved Daten beantwortet werden, mit der Fallback-Antwort „Ich habe keine Information dazu.“ für nicht abgedeckte Fragen. Nutzer können ihre Chatverläufe einsehen, löschen, filtern (mit Suchfunktion), als PDF exportieren oder per E-Mail erhalten. Chatbot-Antworten können per Klick auf ein Sound-Symbol vorgelesen oder per Kopier-Symbol kopiert werden. In einer späteren Version wird die Anwendung barrierefrei erweitert, sodass Benutzer mit Lese- und Schreibschwierigkeiten Fragen per Spracheingabe (via AssembleAPI für Speech-to-Text) stellen und Antworten als Sprachausgabe (via VoiceRSS für Text-to-Speech) erhalten können, während das LLM weiterhin Text-Input und -Output verwendet. Die Anwendung demonstriert Konzepte verteilter Systeme wie API-Integration, Datenpersistenz, RAG-basierte Datenverarbeitung und Echtzeit-Kommunikation.

Aufgabenstellung

Entwicklung einer verteilten Anwendung, die heterogene Datenquellen integriert, Benutzerinteraktionen ermöglicht und die Prinzipien des Moduls „Entwicklung verteilter Anwendungen“ umsetzt, mit einer späteren Erweiterung für Barrierefreiheit.

Projektziele

Die Projektziele sind spezifisch, messbar, erreichbar, relevant und terminiert (SMART):

Spezifisch:
Entwicklung einer Webanwendung mit einem RAG-basierten Chatbot, das Fragen zu Bundesliga-Daten beantwortet, und Funktionen für Chatverlauf-Management, PDF-Export und E-Mail-Versand.

Messbar:
Funktionale Anwendung mit mindestens drei API-Endpunkten (Spieltage, Ergebnisse, Tabellen), Chatbot mit mindestens einem beantwortbaren Fragetyp (Text), RAG mit

mindestens 90 % Trefferquote bei relevanten Daten, Suchfunktion für Chatverläufe und Exportfunktion.

Erreichbar:
Nutzung etablierter Technologien (FastAPI, React, MongoDB), Gemini 1.5 Flash API und einer Vektordatenbank (z. B. Pinecone oder Milvus).

Relevant:
Demonstriert API-Integration, verteilte Datenverarbeitung, RAG und Benutzerinteraktion, passend zum Modul.

Terminiert:
Fertigstellung der Basisversion bis Mitte August 2025 (ca. 10 Wochen).

Muss-Ziele

API-Anbindung, ETL-Prozess und RAG:
Abruf von Spieltagen, Ergebnissen und Tabellenständen über eine externe API (OpenLigaDB), Transformation der Daten in ein LLM-kompatibles Format, Umwandlung in Vektoren und Speicherung in einer Vektordatenbank für semantische Suche, gespeichert in MongoDB.

Chatbot-Integration mit RAG:
Einbindung des Gemini 1.5 Flash LLMs, das Fragen zu Spieldaten basierend auf RAG-retrieved Daten und einem System-Prompt, in dem die Datenquellen mit Beschreibungen zu diesen Datenquellen und Nutzerfrage eingegeben werden, beantwortet, mit Fallback-Antwort „Ich habe keine Information dazu.“ für nicht abgedeckte Fragen.

Chatverlauf-Management:
Speicherung von Nutzerfragen und LLM-Antworten in MongoDB, mit Funktionen zum Anzeigen, Löschen und Filtern (inkl. Suchfunktion) im React-Frontend.

Sprachausgabe und Kopierfunktion:
Implementierung eines Sound-Symbols zur Vorlesung von LLM-Antworten und eines Kopier-Symbols zum Kopieren von Antworten in die Zwischenablage.

PDF-Export und E-Mail-Versand:
Export von Chatverläufen als PDF und Versand per E-Mail an den Nutzer.

Kann-Ziele

Barrierefreie Interaktion (spätere Version):
Integration von Speech-to-Text (AssembleAPI) für Spracheingabe von Fragen und Text-to-Speech (VoiceRSS) für Sprachausgabe von Antworten, um Nutzern mit Lese- und Schreibschwierigkeiten den Zugang zu ermöglichen.

Echtzeit-Updates (Basisversion oder später):

Nutzung von WebSockets für Live-Updates von Spielständen während eines Spieltages, falls die API Echtzeitdaten bereitstellt.

Typische Anwendung

Die Anwendung richtet sich an Fußballfans, Sportjournalisten, Analysten und in einer späteren Version an Nutzer mit Lese- und Schreibschwierigkeiten.

Typische Anwendungsfälle:

Fan:
Ein Nutzer fragt den Chatbot: „Welche Position hat Eintracht Frankfurt in der Tabelle?“ Der RAG-Mechanismus liefert relevante Spieltermine, und die Antwort wird präzise generiert.

Interaktion:
Ein Nutzer klickt auf das Sound-Symbol, um die Antwort „Eintracht Frankfurt ist auf der dritten Position in der Tabelle“ vorgelesen zu bekommen, oder kopiert sie per Klick.

Barrierefreier Zugang (spätere Version):
Ein Nutzer mit Behinderung spricht die Frage „Wie viele Punkte hat Eintracht Frankfurt bis heute erzielt?“ ins Mikrofon (via AssembleAPI), und die RAG-basierte Antwort wird als Sprachausgabe (via VoiceRSS) wiedergegeben.

Zeitplan

Zeitraum: 8 Wochen (Mitte Juni bis Mitte August 2025)
Ziele:

- Ca. 75 % der Ideen umsetzen.
- Beide Personen beteiligen sich an Backend und Frontend.
- Dokumentation, Tests und Architekturdiagramme erstellen.

Woche 1: Setup und Planung
Person 1 (Abdal Ahmad):

- Backend: Einrichtung von FastAPI in VS Code, Test der OpenLigaDB-API für Spieltage und Ergebnisse.
- Frontend: Recherche zur Web Speech API für Sprachausgabe.

Person 2 (Kamal Badawi):

- Backend: Einrichtung von MongoDB, Entwurf des MongoDB-Schemas für Chatverläufe und API-Daten.
- Frontend: Einrichtung von React mit Tailwind CSS in VS Code, Erstellung eines grundlegenden Chatbot-Interface-Mockups.

Gemeinsame Aufgaben:

- Erstellung eines Use-Case-Diagramms (z. B. Nutzerfragen, Chatverlauf, Export).

- Erstellung eines Komponentendiagramms (Frontend, Backend, MongoDB, Vektordatenbank, APIs).
- Einrichtung des GitHub-Repositories, Start des README in Word.

Meilenstein:
- Entwicklungsumgebung funktionsfähig, API-Zugang getestet, Architekturdiagramme fertig.

Woche 2: API-Anbindung und Frontend-Grundlagen

Person 1 (Abdal Ahmad):
- Backend: Implementierung der API-Anbindung (OpenLigaDB) für Spieltage und Ergebnisse mit FastAPI-Endpunkten.
- Frontend: Entwicklung der Chatbot-UI mit Eingabefeld in React.

Person 2 (Kamal Badawi):
- Backend: Setup einer Vektordatenbank und Test der Verbindung, Start des ETL-Prozesses (Extract: API-Daten).
- Frontend: Integration der API-Endpunkte im Frontend (Fetch-Daten mit Axios für Tabellen), Styling der Chatbot-UI mit Tailwind CSS.

Gemeinsame Aufgaben:
- Abstimmung zur Datenstruktur (JSON-Format) via Discord.
- Aktualisierung der Dokumentation (README, Swagger für FastAPI).

Meilenstein:
- API-Daten abrufbar, grundlegende UI fertig, Vektordatenbank eingerichtet.

Woche 3: ETL-Prozess und RAG-Grundlagen
Person 1 (Abdal Ahmad):
- Backend: Transform-Schritt des ETL-Prozesses, Tests.
- Frontend: Implementierung der Chatverlauf-Anzeige in React, Fortsetzung der Frontend-Tests.

Person 2 (Kamal Badawi):
- Backend: Load-Schritt des ETL-Prozesses (MongoDB und Vektordatenbank), Tests.
- Frontend: Implementierung der Chatverlauf-Filter- und Suchfunktionen, Styling der UI mit Tailwind CSS.

Gemeinsame Aufgaben:
- Abstimmung zur RAG-Integration (Datenformat) via Discord.
- Dokumentation des ETL-Prozesses und der UI-Komponenten in Word.

Meilenstein:
- ETL-Prozess funktional, Chatverlauf-Funktionen implementiert.

Woche 4: RAG und Chatbot-Integration
Person 1 (Abdal Ahmad):
- Backend: Integration von RAG: Semantische Suche in der Vektordatenbank.

- Frontend: Implementierung der Kopierfunktion, Verbindung des Chatbot-Interface mit Backend-Endpunkt (via Axios).

Person 2 (Kamal Badawi):
- Backend: Übergabe der RAG-Daten an Gemini 1.5 Flash, Implementierung des System-Prompts.
- Frontend: Implementierung der Sprachausgabe per Symbol-Klick.

Gemeinsame Aufgaben:
- Test der Chatbot-Interaktion (Fragen und Antworten).
- Dokumentation des RAG-Workflows und Chatbot-Funktionen.

Meilenstein:
- RAG-basierter Chatbot funktional, Sprachausgabe (Klickbutton) und Kopierfunktion (Klickbutton) implementiert.

Woche 5: Chatverlauf-Management und Export
Person 1 (Abdal Ahmad):
- Backend: Implementierung der Chatverlauf-Speicherung in MongoDB, E-Mail-Versand mit smtplib (Gmail).
- Frontend: UI für PDF-Export (Klickbutton), Frontend-Tests.

Person 2 (Kamal Badawi):
- Backend: PDF-Export mit pdfkit oder ähnlicher Bibliothek.
- Frontend: Implementierung der Löschfunktion (Klickbutton) für Chatverläufe, UI (Klickbutton) für E-Mail-Versand.

Gemeinsame Aufgaben:
- Test der Export- und E-Mail-Funktionen.
- Dokumentation der Export-Funktionen.

Meilenstein:
- Chatverlauf-Management und Exportfunktionen vollständig implementiert.

Woche 6: WebSockets und Optimierung (optional bei verfügbarer Zeit)
Person 1 (Abdal Ahmad):
- Backend: Implementierung von WebSockets in FastAPI für Echtzeit-Updates.
- Frontend: Optimierung der UI-Performance (z. B. Lazy Loading), Frontend-Tests.

Person 2 (Kamal Badawi):
- Backend: Optimierung der ETL- und RAG-Performance (z. B. Caching).
- Frontend: Integration von WebSockets im Frontend für Echtzeit-Updates.

Gemeinsame Aufgaben:
- Test der Echtzeit-Updates.
- Aktualisierung der Dokumentation (WebSockets, Optimierungen) in Word.

Meilenstein:
- WebSockets implementiert, Performance optimiert.

Woche 7: Testing und Dokumentation
In einer späteren Version der App werden die Tests mit PyTest und Jest automatisiert.
Person 1 (Abdal Ahmad):

- Backend: Manuelles Testen des Backends (API, RAG), API-Tests mit Swagger (FastAPI).
- Frontend: Frontend-Dokumentation (React-Komponenten).

Person 2 (Kamal Badawi):

- Backend: Backend-Dokumentation (ETL, API-Endpunkte).
- Frontend: Manuelles Testen des Frontends (UI, Sprachausgabe, Kopierfunktion, E Mail-Versand, PDF-Export).

Gemeinsame Aufgaben:

- Erstellung des Projektberichts (Architektur, Herausforderungen, Lösungen) in Word/PDF.
- Feinschliff der Architekturdiagramme.

Meilenstein:

- Tests abgeschlossen, Dokumentation weitgehend fertig, Diagramme finalisiert.

Woche 8: Präsentation und Abschluss
Person 1 (Abdal Ahmad):

- Backend: Finale Optimierungen und ggf. Fehlerbehebungen im Backend.
- Frontend: Vorbereitung der Frontend-Demo.

Person 2 (Kamal Badawi):

- Backend: Vorbereitung der Backend-Demo.
- Frontend: Finale Optimierungen und ggf. Fehlerbehebungen im Frontend.

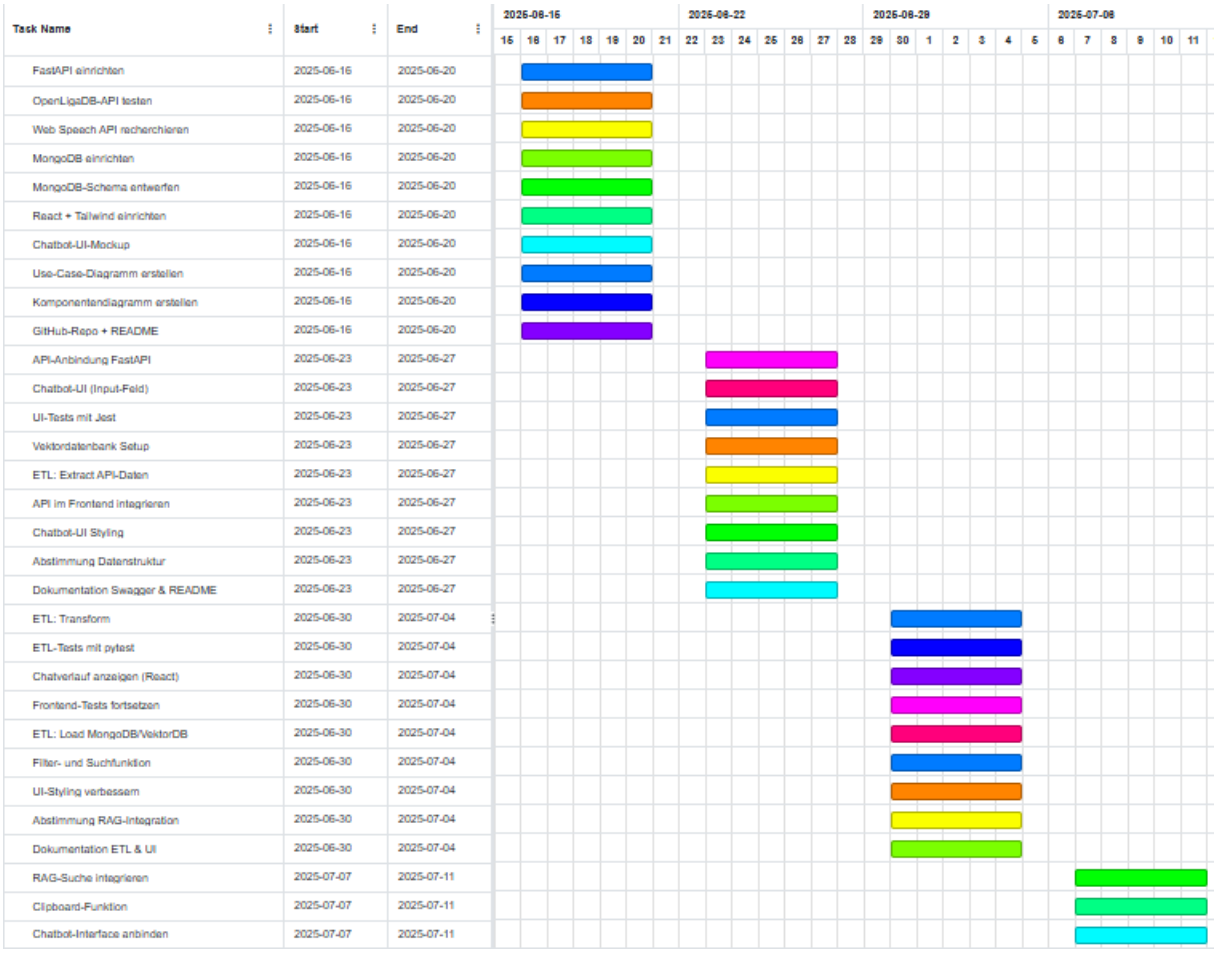
Gemeinsame Aufgaben:

- Erstellung der Präsentation (10-15 Minuten, mit Live-Demo).
- Finale Tests der gesamten Anwendung.
- Abschluss der Dokumentation (README, Projektbericht) in Word/PDF.

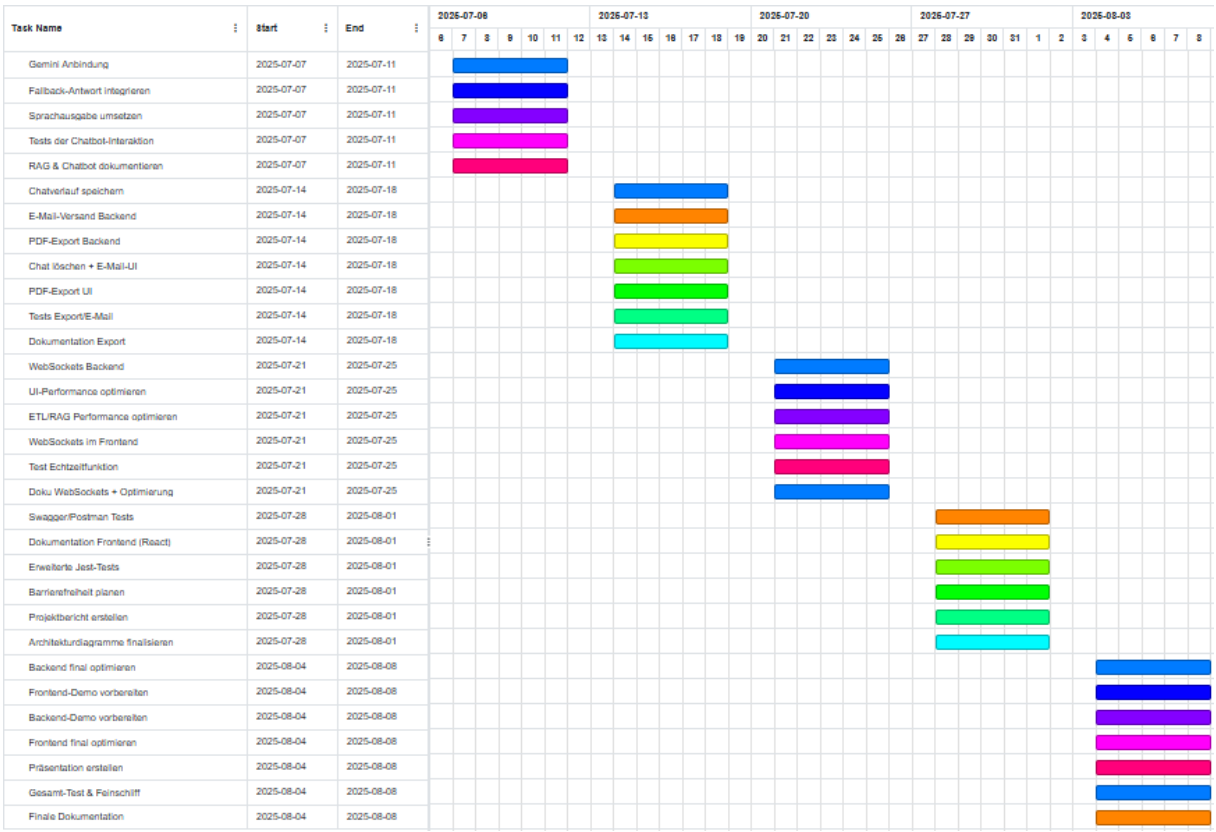
Meilenstein:

- Projekt abgeschlossen, Präsentation und Live-Demo bereit.

Gantt Diagramm



Task Name	Start	End
FastAPI einrichten	2025-06-16	2025-06-20
OpenLigaDB-API testen	2025-06-16	2025-06-20
Web Speech API recherchieren	2025-06-16	2025-06-20
MongoDB einrichten	2025-06-16	2025-06-20
MongoDB-Schema entwerfen	2025-06-16	2025-06-20
React + Tailwind einrichten	2025-06-16	2025-06-20
Chatbot-UI-Mockup	2025-06-16	2025-06-20
Use-Case-Diagramm erstellen	2025-06-16	2025-06-20
Komponentendiagramm erstellen	2025-06-16	2025-06-20
GitHub-Repo + README	2025-06-16	2025-06-20
API-Anbindung FastAPI	2025-06-23	2025-06-27
Chatbot-UI (Input-Feld)	2025-06-23	2025-06-27
UI-Tests mit Jest	2025-06-23	2025-06-27
Vektordatenbank Setup	2025-06-23	2025-06-27
ETL: Extract API-Daten	2025-06-23	2025-06-27
API im Frontend integrieren	2025-06-23	2025-06-27
Chatbot-UI Styling	2025-06-23	2025-06-27
Abstimmung Datenstruktur	2025-06-23	2025-06-27
Dokumentation Swagger & README	2025-06-23	2025-06-27
ETL: Transform	2025-06-30	2025-07-04
ETL-Tests mit pytest	2025-06-30	2025-07-04
Chatverlauf anzeigen (React)	2025-06-30	2025-07-04
Frontend-Tests fortsetzen	2025-06-30	2025-07-04
ETL: Load MongoDB/VektorDB	2025-06-30	2025-07-04
Filter- und Suchfunktion	2025-06-30	2025-07-04
UI-Styling verbessern	2025-06-30	2025-07-04
Abstimmung RAG-Integration	2025-06-30	2025-07-04
Dokumentation ETL & UI	2025-06-30	2025-07-04
RAG-Suche integrieren	2025-07-07	2025-07-11
Clipboard-Funktion	2025-07-07	2025-07-11
Chatbot-Interface anbinden	2025-07-07	2025-07-11



Task Name	Start	End
Gemini Anbindung	2025-07-07	2025-07-11
Fallback-Antwort integrieren	2025-07-07	2025-07-11
Sprachausgabe umsetzen	2025-07-07	2025-07-11
Tests der Chatbot-Interaktion	2025-07-07	2025-07-11
RAG & Chatbot dokumentieren	2025-07-07	2025-07-11
Chatverlauf speichern	2025-07-14	2025-07-18
E-Mail-Versand Backend	2025-07-14	2025-07-18
PDF-Export Backend	2025-07-14	2025-07-18
Chat löschen + E-Mail-UI	2025-07-14	2025-07-18
PDF-Export UI	2025-07-14	2025-07-18
Tests Export/E-Mail	2025-07-14	2025-07-18
Dokumentation Export	2025-07-14	2025-07-18
WebSockets Backend	2025-07-21	2025-07-25
UI-Performance optimieren	2025-07-21	2025-07-25
ETL/RAG Performance optimieren	2025-07-21	2025-07-25
WebSockets im Frontend	2025-07-21	2025-07-25
Test Echtzeitfunktion	2025-07-21	2025-07-25
Doku WebSockets + Optimierung	2025-07-21	2025-07-25
Swagger/Postman Tests	2025-07-28	2025-08-01
Dokumentation Frontend (React)	2025-07-28	2025-08-01
Erweiterte Jest-Tests	2025-07-28	2025-08-01
Barrierefreiheit planen	2025-07-28	2025-08-01
Projektbericht erstellen	2025-07-28	2025-08-01
Architekturdiagramme finalisieren	2025-07-28	2025-08-01
Backend final optimieren	2025-08-04	2025-08-08
Frontend-Demo vorbereiten	2025-08-04	2025-08-08
Backend-Demo vorbereiten	2025-08-04	2025-08-08
Frontend final optimieren	2025-08-04	2025-08-08
Präsentation erstellen	2025-08-04	2025-08-08
Gesamt-Test & Feinschliff	2025-08-04	2025-08-08
Finale Dokumentation	2025-08-04	2025-08-08

Zusammenarbeit (Tools)

Für die erfolgreiche Zusammenarbeit im Projekt „Bundesliga-ChatBot“ werden verschiedene Tools und Technologien eingesetzt, die unterschiedliche Bereiche der Entwicklung, Kommunikation und Organisation abdecken.

Versionskontrolle: Zur Versionskontrolle kommt **GitHub** zum Einsatz. Es dient der zentralen Codeverwaltung und ermöglicht über Pull Requests, Branching und Issue-Tracking eine strukturierte und nachvollziehbare Entwicklung im Team.

Kommunikation: Die Kommunikation erfolgt primär über **Discord**. Dort werden laufende Absprachen getroffen, Feedback ausgetauscht und bei Bedarf Dateien oder Links geteilt.

Dokumentation: Für die Dokumentation wird **Microsoft Word** (Google Docs) genutzt, um den Projektbericht, Protokolle, den Zeitplan sowie Architektur- und Technikkonzepte festzuhalten. Zusätzlich sorgt **Swagger**, integriert über FastAPI, für eine automatische und interaktive API-Dokumentation, die sowohl der Übersicht als auch dem Testen der Endpunkte dient.

Entwicklung: Die gesamte Entwicklung erfolgt in der gemeinsam genutzten IDE **Visual Studio Code** (VS Code). Sowohl Backend- als auch Frontend-Komponenten werden dort entwickelt, was durch passende Erweiterungen und Tooling für **Python, React und TailwindCSS** effizient unterstützt wird.

Testing: Im Bereich Testing wird in einer späteren Version für das Backend **pytest** verwendet. Es ermöglicht umfassende Tests für API-Funktionalitäten, ETL-Prozesse und RAG-Komponenten. Für das Frontend kommt **Jest** in einer späteren Version zum Einsatz, um React-Komponenten und UI-Interaktionen automatisiert zu testen.

Systemarchitektur: Zur Veranschaulichung der Systemarchitektur und funktionalen Abläufe werden Diagramme mithilfe von **Visual Paradigm** oder alternativ **drawi.io** erstellt. Dazu gehören insbesondere Use-Case-Diagramme, Komponentenübersichten, ER-Modell und Architekturdiagramme.

Offene Fragen

Verfügbarkeit und Stabilität der Datenquelle (OpenLigaDB):

- Gibt es ausreichende Dokumentation und eine zuverlässige Verfügbarkeit der OpenLigaDB-API über den gesamten Projektzeitraum?
- Ist die Datenaktualisierung in Echtzeit oder nur in Intervallen verfügbar?

Kosten und Nutzungslimit externer Dienste:

- Gibt es Nutzungseinschränkungen oder Kosten bei der Verwendung der Gemini 1.5 Flash API, AssembleAPI (Speech-to-Text) und VoiceRSS (Text-to-Speech)?
- Müssen wir kostenpflichtige Konten anlegen oder reicht die kostenlose Nutzung für das Projekt?

Datenschutz und Benutzerverwaltung:

- Werden personenbezogene Daten gespeichert (z. B. bei E-Mail-Versand)? Falls ja, wie stellen wir DSGVO-Konformität sicher?
- Wird eine Nutzeranmeldung (z. B. mit Login-System) notwendig sein, oder bleibt die Anwendung anonym?

Barrierefreiheit und Browserkompatibilität:

- Wie stellen wir sicher, dass die Web Speech API und weitere barrierefreie Funktionen in allen gängigen Browsern (Chrome, Firefox, Safari) funktionieren?
- Gibt es Alternativen, falls APIs (z. B. AssembleAPI) nicht zuverlässig oder browserunabhängig funktionieren?

Vektordatenbankwahl und -hosting:

- Welche Vektordatenbank (z. B. Pinecone, Milvus, Weaviate) eignet sich am besten für das Projekt, auch im Hinblick auf kostenlose Nutzungsmöglichkeiten?
- Wo wird die Datenbank gehostet - lokal, cloudbasiert oder über einen Drittanbieter?

Fallback-Strategie bei LLM-Ausfall:

- Was passiert, wenn die Verbindung zur Gemini API temporär nicht verfügbar ist?
- Ist ein alternativer Modus ohne LLM denkbar (z. B. direkte Anzeige strukturierter Daten)?

Logo

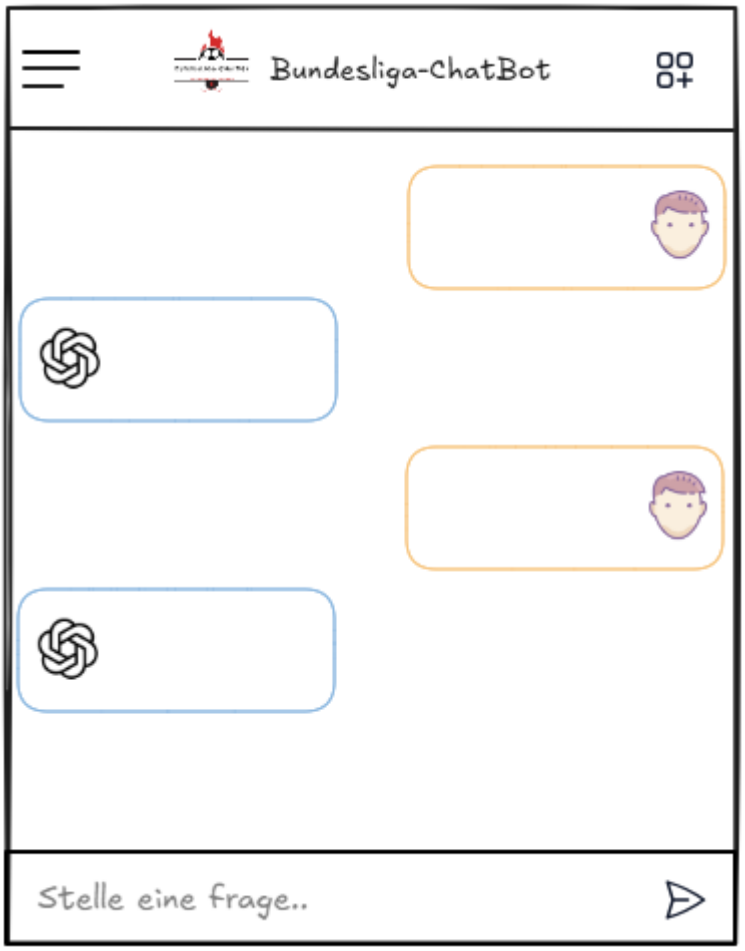
Für unser entwickeltes Projekt wurde das folgende Logo mithilfe des Online-Tools DesignEvo selbst gestaltet. Dabei orientiert sich die Farbwahl bewusst an den typischen Farben des Bundesliga-Logos, um einen sportlichen und wiedererkennbaren Charakter zu unterstreichen.

Mockup

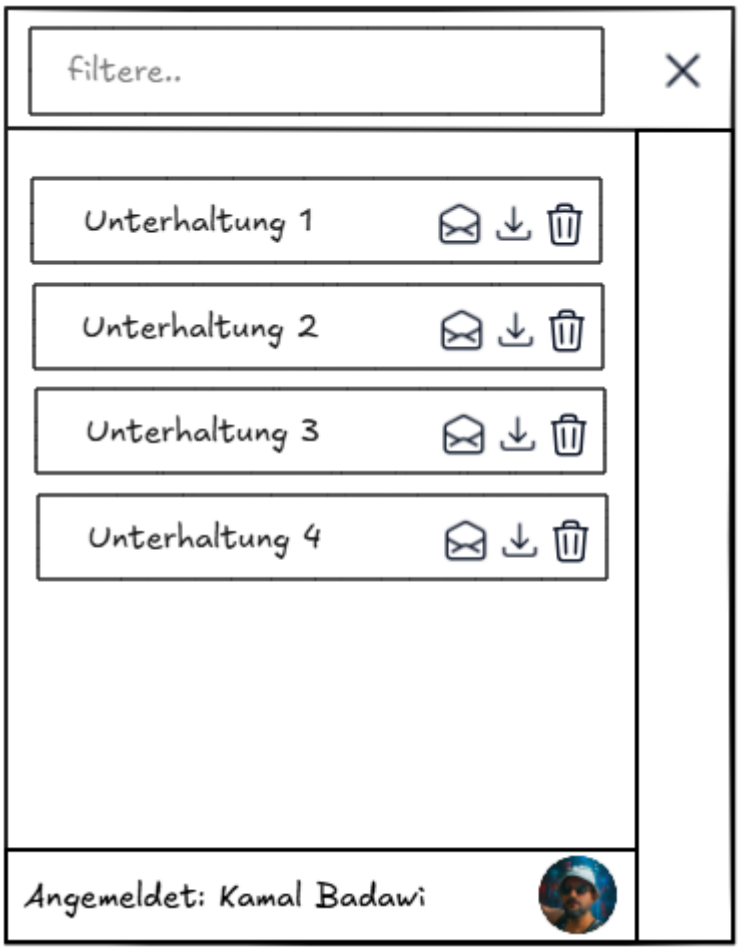
Für die Gestaltung der Mockups kam das Tool Excalidraw zum Einsatz. Die im Projekt verwendeten Symbole stammen aus SVG-Sammlungen von Heroicons sowie Icons8, um ein modernes und konsistentes Design zu gewährleisten.

Unsere grafische Benutzeroberfläche (GUI) gliedert sich in zwei zentrale Ansichten, die eine benutzerfreundliche und effiziente Interaktion ermöglichen:

- **Nachrichtenansicht:** In dieser Ansicht kann der Nutzer dem Chatbot Fragen stellen, welche in Echtzeit von einem leistungsstarken Sprachmodell (LLM) beantwortet werden. Der Fokus liegt hierbei auf einer intuitiven und dialogorientierten Kommunikation.



- **Konversationsansicht**: Diese bietet eine strukturierte Übersicht über sämtliche bisher geführten Unterhaltungen. Nutzerinnen und Nutzer können Konversationen gezielt filtern, per Mail versenden, als PDF-Datei exportieren oder dauerhaft löschen, um die Übersichtlichkeit und persönliche Organisation der Inhalte zu optimieren.



Ergebnisdokumentation

Technische Dokumentation:
Swagger dokumentiert FastAPI-Endpunkte (Spieltage, Ergebnisse, Tabellen). Der Code (FastAPI, React) wird in GitHub mit Kommentaren und einer README-Datei verwaltet, die Projektstruktur und Installation beschreibt.

Projektbericht:
Ein Word-Bericht (als PDF exportiert) fasst SMART-Ziele, Systemarchitektur (FastAPI, React, MongoDB, Vektordatenbank), Zeitplan (8 Wochen, Juni-August 2025), Herausforderungen und die geplante Barrierefreiheit zusammen. Wöchentliche Updates, finalisiert in Woche 7/8.

Architekturdiagramme: Use-Case- und Komponentendiagramme (draw.io oder Visual Paradigm) visualisieren System und Interaktionen.

Testdokumentation (Optional): pytest (Backend: API, ETL, RAG) und Jest (Frontend: UI, Chatverlauf) Tests werden in Word protokolliert, mit Metriken (z. B. 90 % RAG-Trefferquote).

Präsentation: Endpräsentation mit Live-Demo, Screenshots, Diagrammen und Testergebnissen, basierend auf dem Bericht.

Visuelles Design: Logo (DesignEvo, Bundesliga-Farben) und GUI-Mockups (Excalidraw, Heroicons/Icons8) in Bericht und Präsentation integriert.

Die Dokumentation wird wöchentlich aktualisiert, um Fortschritte zu sichern, und liefert am Ende eine klare, vollständige Darstellung der technischen und organisatorischen Ergebnisse, inklusive Planung für die spätere barrierefreie Version.