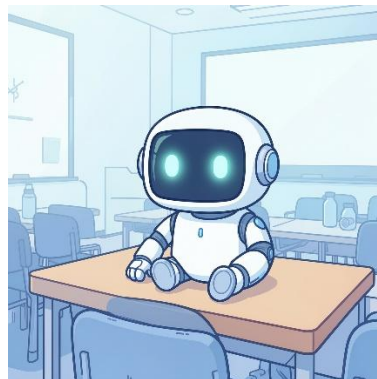




Projektbericht



Misty II als KI-gestützter Sprach-Coach im Bildungskontext

1 Problemstellung

Mündliche Kommunikation und Rhetorik zählen zu den zentralen Kompetenzen im schulischen und beruflichen Kontext. Forschungsbefunde zeigen, dass frühzeitig geförderte mündliche Sprachfähigkeiten nicht nur spätere Leseleistungen, sondern auch kognitive Fähigkeiten und sozioemotionale Kompetenzen positiv beeinflussen (Girolametto & Weitzman, 2023). Dennoch ist individuelles Feedback im Unterricht kaum skalierbar: Eine Lehrkraft kann nicht jede Schülerin und jeden Schüler gleichzeitig beobachten und gezielte Rückmeldung zu Füllwörtern oder Sprechtempo geben. Automatisierte Systeme existieren zwar in Ansätzen, sind jedoch häufig auf Cloud-Dienste angewiesen oder nicht für den Unterrichtseinsatz ausgelegt.

Das vorliegende Projekt begegnet dieser Lücke mit Misty II als interaktivem Sprach-Coach, der Schülerinnen und Schülern unmittelbares, datenschutzkonformes Feedback gibt. Belpaeme et al. (2018) zeigen, dass soziale Roboter im Bildungskontext Motivation und Interaktionsbereitschaft stärker fördern als rein digitale Alternativen. Rasouli et al. (2022) ergänzen, dass die wahrgenommene Verspieltheit sozialer Roboter soziale Angst reduzieren kann, was einen entscheidenden Vorteil für Kinder mit Präsentationsangst darstellt, die zunächst angstfrei vor einem Roboter üben und so schrittweise Selbstsicherheit aufbauen können.

2 Anforderungsanalyse

Die Analyse von Anforderungen bildet den Ausgangspunkt jeder Entwicklung digitaler Anwendungen, da sie sicherstellt, dass das entstehende System den tatsächlichen Bedürfnissen der Nutzenden entspricht. Sie umfasst die Festlegung der Ziele und Aufgaben des Systems und legt damit die Grundlage für alle weiteren Entwicklungsschritte (Broy und Kuhrmann 2021). Im vorliegenden Projekt wurden zu diesem Zweck zwei Nutzungsszenarien definiert, aus denen anschließend die funktionalen und nicht-funktionalen Anforderungen abgeleitet wurden.

In Szenario A stellt sich eine Schülerin oder ein Schüler vor den Roboter Misty II und hält eine kurze freie Rede oder ein Referat. Durch Drücken des Fußsensors startet die Aufnahme; Misty bestätigt dies mit dem Satz „Ich höre dir zu“. Nach dem Ende der Rede, signalisiert durch erneutes Drücken des Sensors, analysiert das System die Aufnahme und gibt unmittelbar Feedback: Misty nennt die Anzahl erkannter Füllwörter, bewertet das Sprechtempo und zeigt einen passenden Gesichtsausdruck.

In Szenario B beobachtet eine Lehrkraft den Verlauf mehrerer Sessions über ein webbasiertes Dashboard, das in Echtzeit Statusinformationen, Füllwortanzahl, Sprechtempo, Transkript und Sessionverlauf anzeigt. Füllwörter können über das Dashboard individuell angepasst werden, etwa je nach Altersstufe oder Unterrichtskontext.

Aus diesen beiden Szenarien ergeben sich folgende Anforderungen:

Funktionale Anforderungen:

- Sprache lokal transkribieren
- Definierte Füllwörter erkennen und zählen
- Sprechtempo berechnen und bewerten
- Feedback per Sprache und Gesichtsausdruck ausgeben
- Ergebnisse in Echtzeit auf einem Dashboard darstellen
- Füllwortliste über Dashboard anpassbar machen

Nicht-funktionale Anforderungen:

- Datenschutzkonformität durch vollständig lokale Verarbeitung
- Zuverlässigkeit des Fußsensor-Triggers
- Echtzeitfähigkeit der Dashboard-Aktualisierung

Die Feedback-Kriterien basieren auf einer didaktisch und wissenschaftlich begründeten Logik: Seals und Coppock (2022) zeigen in ihrer Studie, dass übermäßiger Füllwortgebrauch sowohl die Glaubwürdigkeit der Sprechenden als auch das Textverständnis beim Publikum messbar beeinträchtigt. Als wirksame Gegenmaßnahmen identifizieren sie gezieltes, verstärkendes Feedback sowie aktive Intervention zur Reduktion verbaler Füllpausen. Auf dieser Grundlage wurde die Grenze von drei oder mehr Füllwörtern als problematischer Bereich definiert, der eine entsprechende Rückmeldung durch Misty auslöst. Einzelne Füllwörter sind in natürlicher Rede tolerierbar und werden daher im Bereich von null bis zwei als akzeptabel gewertet.

Das Sprechtempo wird im Bereich von 90 bis 150 Wörtern pro Minute als optimal bewertet. Walsh et al. (2021) zeigen in einer Studie mit 62 typisch entwickelten Kindern im Alter von 10 bis 14 Jahren, dass das Sprechtempo bei Kindern mit der Satzlänge variiert und sich im Laufe der Entwicklung durch zunehmende Effizienz in motorischen und kognitiv-linguistischen Prozessen steigert. Dieser Befund begründet die Entscheidung, für die Zielgruppe Kinder und Jugendliche eine niedrigere Untergrenze von 90 Wörtern pro Minute anzusetzen als für Erwachsene üblich. Aus der Kombination beider Parameter ergibt sich eine 2×2-Matrix, die den Gesichtsausdruck von Misty bestimmt: Bei wenigen Füllwörtern und optimalem Tempo zeigt Misty ein positives Gesicht, bei vielen Füllwörtern und schlechtem Tempo ein trauriges Gesicht, die beiden Mischfälle werden mit einem neutralen Ausdruck quittiert.

3 Entwicklungsprozess

Die Entwicklung des Misty Sprach-Coaches folgte einem agilen, iterativen Ansatz. Anstatt alle Funktionen vorab vollständig zu planen, wurde das System schrittweise aufgebaut: implementiert, getestet, reflektiert und verbessert. Die einzelnen Kernaufgaben wurden dabei nicht sequenziell, sondern häufig miteinander verzahnt bearbeitet (Broy & Kuhrmann, 2021). Fehler und Hindernisse wurden dabei als Lernchancen verstanden, die den Entwicklungsprozess aktiv vorangetrieben haben (Hegele & Stoll, 2026).

Der Entwicklungsverlauf gliederte sich in acht Phasen:

Phase	Bezeichnung	Beschreibung
1	Infrastruktur & Konnektivität	SSH-Verbindung zwischen Ubuntu-Server und Misty II. Erste API-Validierung
2	Audio-Pipeline	Push- vs. Pull-Mechanismus getestet → Pull wegen Stabilität gewählt
3	KI-Integration	Verschiedene Whisper-Modelle getestet und eingebunden → base gewählt (größere Modelle ohne GPU nicht einsetzbar), Füllworterkennung per Regex implementiert
4	Optimierung	Whisper-Modell gecacht, Halluzinationen durch Parameteranpassung behoben
5	Sprechtempo-Analyse	Sprechtempo aus Wortanzahl und echter Aufnahmedauer berechnet.

6	Feedback-Logik	Differenziertes Feedback je nach Kombination aus Füllwörtern und Sprechtempo implementiert, ausgegeben per Sprache und Gesichtsausdruck.
7	Livestream-Modus (verworfen)	RTSP-Stream getestet und bewusst verworfen – zu instabil, kein didaktischer Mehrwert
8	Web-Dashboard	Flask-Webserver mit Jinja2-Templates: SSE Live-Updates, aktuelle Session, Sessionverlauf und anpassbare Füllwortliste.
9	Robustheit	Heartbeat-Thread, Error Handling

Als Ergebnis des iterativen Entwicklungsprozesses entstand ein funktionierender Prototyp: Misty II nimmt Sprechproben auf, erkennt Füllwörter und analysiert das Sprechtempo. Das direkte Feedback per Sprache und Gesichtsausdruck richtet sich an den Sprechenden, während das Web-Dashboard Lehrenden einen Echtzeit-Überblick über Ergebnisse, Sessionverlauf und Füllwortverwaltung bietet.

Zur Versionskontrolle wurde Git in Verbindung mit dem Gitea-Server der Pädagogischen Hochschule Weingarten eingesetzt. Der gesamte Quellcode sowie [Screenshots des Dashboards](https://git.md-phw.de/TiffanyBrugger/Misty-Sprach-Coach) sind öffentlich zugänglich und einsehbar unter: <https://git.md-phw.de/TiffanyBrugger/Misty-Sprach-Coach>

Der entwickelte Livestream-Modus wurde nach ausgiebigen Tests bewusst nicht in den Hauptworkflow integriert. Mehrere Gründe sprachen dagegen: Whisper ist darauf ausgelegt vollständige Audioblöcke zu analysieren, weshalb zu kurze Echtzeit-Fragmente zu deutlich schlechterer Füllworterkennung führen. Darüber hinaus überträgt ein kontinuierlicher Livestream große Datenmengen über das WLAN, was die Verbindung zusätzlich belastet und zu Instabilitäten führte. Letztlich sprach auch die didaktische Perspektive gegen den Livestream-Modus: Da der Nutzer ohnehin auf das Feedback wartet, entsteht kein Mehrwert gegenüber dem Aufnahme-Modus, der zudem eine deutlich bessere Transkriptionsqualität bei gleichem Lernergebnis liefert. Der experimentelle Code ist im Repository unter [livemodus experimentell/](#) einsehbar.

4 Systemarchitektur

Für die Umsetzung des Misty Sprach-Coaches kamen verschiedene Programmiersprachen, Frameworks und Schnittstellen zum Einsatz. Als Programmiersprache diente Python 3.12, das Web-Dashboard wurde mit dem Framework Flask realisiert, für die Sprachtranskription wurde das lokal laufende KI-Modell OpenAI Whisper verwendet, die Kommunikation mit Misty II erfolgte über die Misty Robotics REST-API und WebSocket, und Live-Updates im Dashboard wurden mittels Server-Sent Events (SSE) umgesetzt.

Das Zusammenspiel dieser Komponenten lässt sich anhand einer Client-Server-Architektur mit vier Schichten beschreiben. Die Hardware-Schicht bildet Misty II, der gleichzeitig als Eingabe- und Ausgabegerät fungiert. Der Fuß-Sensor dient als physischer Auslöser: Ein erster Fußdruck sendet ein Ereignis per WebSocket-Protokoll an den Ubuntu-Server und startet die Aufnahme, ein zweiter Fußdruck stoppt sie. Das Mikrofon nimmt die Sprache als WAV-Datei auf. Für die Übertragung der Audiodatei wurde ein Pull-Mechanismus implementiert, bei dem der Server die Datei aktiv von Misty abrufen, anstatt dass diese eigenständig hochgeladen wird. Diese Architekturentscheidung erwies sich gegenüber einem Push-Ansatz als robuster gegen-

über Verbindungsschwankungen. Das Feedback erfolgt über den Lautsprecher mittels Text-to-Speech sowie über das Display durch passende Gesichtsausdrücke.

Die Verarbeitungsschicht läuft auf einem Ubuntu-Server und umfasst vier Python-Skripte: `config.py` enthält die zentrale IP-Konfiguration, `start_coaching.py` koordiniert den gesamten Programmablauf inklusive Heartbeat-Thread zur Verbindungsüberwachung: vom Warten auf den Fuß-Sensor über die Aufnahme und den Download der Audiodatei bis hin zum Anstoßen der Analyse und der Rückmeldung an Misty. `analyse.py` übernimmt die KI-gestützte Transkription sowie die Füllworterkennung und die Sprechtempo-Berechnung. Bei der Füllworterkennung wurden zwei Besonderheiten berücksichtigt: Whisper wird beim Transkribieren über einen `initial_prompt` mit der aktuellen Füllwortliste versorgt, sodass das Modell diese Wörter gezielt erkennt und nicht aufgrund seines Trainings auf verständliche Texte ignoriert. Darüber hinaus wurde die Zähllogik so implementiert, dass ausschließlich vollständige Wörter erfasst werden, sodass beispielsweise „ähm“ nicht versehentlich als Instanz von „äh“ gewertet wird, und dass Groß- und Kleinschreibung vereinheitlicht wird, da Whisper Füllwörter je nach Kontext in unterschiedlicher Schreibweise ausgibt. Das Sprechtempo wird aus der Wortanzahl und der echten Aufnahmedauer berechnet und in Wörtern pro Minute angegeben. `dashboard.py` stellt einen Flask-Webserver bereit und nutzt Jinja2-Templates zur dynamischen HTML-Generierung.

Als Datenschicht dienen zwei JSON-Dateien: `session_daten.json` als Verbindungsglied zwischen Analyse und Dashboard sowie `fuellwoerter.json` als anpassbare Füllwortliste.

Die Darstellungsschicht bildet das Web-Dashboard im Browser, das Live-Updates per Server-Sent Events empfängt und Lehrenden Ergebnisse, Sessionverlauf und Füllwortverwaltung in Echtzeit anzeigt.

5 Einschränkungen & Ausblick

Das entwickelte System erfüllt die gesetzten Projektziele als funktionierender Prototyp. Im Rahmen des Moduls wurden folgende Einschränkungen bewusst in Kauf genommen: Die Sprachunterstützung ist auf Deutsch beschränkt. Das lokal laufende Whisper base-Modell liefert keine perfekte Transkriptionsqualität. Ein größeres Modell wäre genauer, ist auf dem Server ohne GPU jedoch nicht einsetzbar. Zudem ändert sich Mistys IP-Adresse bei jedem Neustart und muss manuell aktualisiert werden. Der entwickelte Livestream-Modus wurde bewusst nicht in den Hauptworkflow integriert, da Whisper nicht für Echtzeit-Verarbeitung ausgelegt ist und die WLAN-Verbindung zu instabil war.

Für eine Weiterentwicklung bieten sich folgende Ansätze an: Der Einsatz eines größeren Whisper-Modells auf leistungstärkerer Hardware würde die Transkriptionsqualität deutlich verbessern. Eine persistente Datenspeicherung über Sessions hinaus in Verbindung mit einer Benutzerverwaltung im Dashboard würde langfristige Lernfortschritte mehrerer Schüler sichtbar und vergleichbar machen. Darüber hinaus wäre eine Erweiterung der Analysekriterien denkbar, etwa durch die Messung von Lautstärke und Sprechpausen als zusätzliche Feedback-Dimensionen.

6 Fachliteratur, verwendete Tools/Bibliotheken & KI

Fachliteratur

- Belpaeme, T., Kennedy, J., Ramachandran, A., Scassellati, B., & Tanaka, F. (2018). Social robots for education: A review. *Science Robotics*, 3(21).
- Broy, M., & Kuhrmann, M. (2021). *Einführung in die Softwaretechnik*. Springer Berlin Heidelberg. <https://doi.org/10.1007/978-3-662-50263-1>
- Hegele, Y., & Stoll, A. (Hrsg.). (2026). *Agiles Arbeiten im öffentlichen Sektor: Erfahrungen in Deutschland, Österreich und der Schweiz*. Springer VS.
- Rasouli, S., Gupta, G., Nilsen, E., & Dautenhahn, K. (2022). Potential applications of social robots in robot-assisted interventions for social anxiety. *International Journal of Social Robotics*, 14(5), 1–32. <https://doi.org/10.1007/s12369-021-00851-0>
- Seals, D. R., & Coppock, M. E. (2022). We, um, have, like, a problem: excessive use of fillers in scientific speech. *Advances in Physiology Education*, 46(4), 615–620.
- Girolametto, L., & Weitzman, E. (2023). Early childhood education language environments: considerations for research and practice. *Frontiers in Psychology*, 14.
- Walsh, B., Smith, A., & Weber-Fox, C. (2021). Speech Rate Varies With Sentence Length in Typically Developing Children. *Journal of Speech, Language, and Hearing Research*, 64(6), 2385–2391.

Verwendete Tools und Bibliotheken

Die vollständige Liste aller verwendeten Pakete und Versionsnummern ist in der [requirements.txt](#) im [Git-Repository](#) einsehbar. Die wichtigsten eingesetzten Tools und Bibliotheken sind:

- OpenAI Whisper (openai-whisper) (Version 20250625): Lokale Sprachtranskription
- Flask (Version 3.1.3): Webserver für das Dashboard
- requests (Version 2.32.5): HTTP-Kommunikation mit Misty API
- websocket-client (Version 1.9.0): WebSocket-Verbindung für Bumper-Ereignisse
- Misty Robotics API: REST-API zur Robotersteuerung

Verwendung von KI

Im Rahmen dieses Projekts wurden Claude (Anthropic) und ChatGPT (OpenAI) unterstützend zur Code-Strukturierung, zum Debugging und zur sprachlichen Ausarbeitung der Dokumentation eingesetzt. Sämtlicher generierter Code wurde kritisch evaluiert und angepasst. Konzeption, didaktische Einbettung und alle technischen Entscheidungen stellen eine eigenständige Transferleistung des Projektteams dar. OpenAI Whisper wurde zur lokalen Sprachtranskription eingesetzt.