Probearbeit 02.2026

Probearbeit: AI-Powered Messenger-Anrufbeantworter

Kontext & Vision

Ein Nutzer ruft eine Telefonnummer an und bekommt per Sprache eine Zusammenfassung seiner ungelesenen Nachrichten aus einem Messenger-Dienst vorgelesen – inklusive Absender, Zeitpunkt und Inhalt.

Aufgabenstellung

Entwickle einen Prototyp eines sprachgesteuerten Anrufbeantworters, der:

1. Einen eingehenden Telefonanruf entgegennimmt
2. Ungelesene Nachrichten automatisch aus einem Messenger abruft
3. Die Nachrichten per LLM zusammenfasst
4. Die Zusammenfassung als Sprache an den Anrufer zurückgibt
5. Als Container deployed werden kann (inkl. Deployment-Manifest)

Funktionale Anforderungen

Kern-Flow

```
Anruf eingehend
  → Begrüßung ("Du hast X ungelesene Nachrichten...")
  → Automatischer Abruf ungelesener Nachrichten via Messenger-API
  → LLM-Zusammenfassung (Absender, Datum/Uhrzeit, Kerninhalt)
  → Sprachausgabe der Zusammenfassung an den Anrufer
```

Bidirektionale Sprachinteraktion

- Der Anrufer kann per Sprache Rückfragen stellen (z. B. „Erzähl mir mehr über Nachricht 3")
- Die Rückfragen werden sinnvoll verarbeitet und beantwortet
- Fehlerbehandlung bei unverständlichen oder unerwarteten Eingaben

Messenger-Integration

- Anbindung an eine Messenger-API (Empfehlung: Telegram Bot API – geringe Einstiegshürde, kein OAuth, kostenlos, gut dokumentiert. Andere Messenger sind ebenfalls akzeptabel.)
- Automatischer Abruf ungelesener Nachrichten mit Metadaten (Absender, Timestamp, Text)
- Saubere Fehlerbehandlung (keine Nachrichten vorhanden, API nicht erreichbar, etc.)
- Hinweis Telegram: Die Bot API erlaubt nur das Lesen von Nachrichten, die an den Bot gesendet wurden – nicht die eigenen Nachrichten des Nutzers. Das ist für diese Aufgabe völlig ausreichend und so gewollt.

Sprach-Pipeline

- Bidirektionales Audio-Streaming (Spracheingabe & Sprachausgabe)
- Telephony-Anbindung (z. B. Twilio Free-Tier, Vonage Trial o. ä.)

LLM-Zusammenfassung

- Prompt-Design für strukturierte Zusammenfassung
- Format: Absender → Zeitpunkt → Zusammenfassung des Inhalts
- Umgang mit mehreren Nachrichten (Priorisierung / Gruppierung)

Technische Anforderungen

Code & Architektur

- Programmiersprache: frei wählbar
- Klare Trennung der Komponenten
- Sinnvolle Fehlerbehandlung und Logging
- README mit Setup-Anleitung und Architektur-Übersicht

Containerisierung & Deployment

- **Dockerfile** für den Service
- **Deployment-Manifest** (z. B. Kubernetes YAML, Helm Chart oder Kustomize)
- **Lokales Kubernetes-Cluster** (z. B. Kind, K3s o. ä.)
- Konfiguration über Umgebungsvariablen (API-Keys, Messenger-Tokens, etc.)
- Health-Check / Readiness-Probe

Rahmenbedingungen

- **Zeitrahmen:** 2-3 Arbeitstage nach Erhalt der Aufgabenstellung
- **Abgabe:** Git-Repository (GitHub/GitLab)
  - Vollständiger Code, Dockerfile, Deployment-Manifest und README
  - **Die Arbeitsschritte müssen in sinnvollen, nachvollziehbaren Commits ersichtlich sein** (keine einzelne „Initial commit"-Abgabe – wir möchten den Entwicklungsprozess und die Arbeitsweise sehen)
- **Demo:** Kurze Live-Demo (ca. 15 Min.) mit anschließender Besprechung der Architektur und Entscheidungen (ca. 15 Min.)
- **API-Keys:** Ein LLM-Key wird von uns gestellt. Das Telegram-Bot-Token (bzw. Zugangsdaten für den gewählten Messenger) erstellt der Bewerber eigenständig.
- **Kubernetes:** Lokales Cluster, z. B. via Kind oder K3s

Hinweise

- Es geht um einen funktionalen Prototyp, nicht um ein produktionsreifes System.
- Pragmatische Abkürzungen sind erlaubt und erwünscht – bitte dokumentiere, was du in einem Produktivsystem anders machen würdest.
- Der Fokus liegt auf der Gesamtarchitektur, der Messenger-Integration, der LLM-Zusammenfassung und der Sprach-Ein-/Ausgabe – nicht auf der Telefonie-Infrastruktur.
- Zeige, dass du Entscheidungen bewusst triffst und begründen kannst.
- Wenn du an einer Stelle nicht weiterkommst: Gib trotzdem den aktuellen Stand ab. Dokumentiere, wo das Problem liegt, was du bereits versucht hast und warum es nicht funktioniert hat. Das zeigt uns mehr über deine Arbeitsweise als ein fertiges Ergebnis, bei dem der Weg nicht nachvollziehbar ist.
- KI unterstützte Entwicklung ist natürlich erlaubt, du solltest aber begründen können warum du welche Entscheidung akzeptiert hast