

Übungsaufgabe: Multi-Agent Customer Support System

Aufgabenbeschreibung

Entwickle ein Multi-Agent System für automatisierten Kundensupport mit 3 spezialisierten Agents, die zusammenarbeiten, um Kundenanfragen intelligent zu bearbeiten.

Technische Anforderungen

SDK-Vorschläge (freie Wahl):

- OpenAI Agents SDK
- Vercel AI SDK mit Multi-Agent Pattern
- Pydantic AI für strukturierte Agent-Workflows
- Oder andere Frameworks deiner Wahl

System-Architektur

Agent 1: Router Agent

- Klassifiziert eingehende Anfragen in 3 Kategorien
- **Categories:** "technical", "billing", "general"
- **Output:** Kategorie + weitergeleitete Anfrage

Agent 2: Knowledge Agent

- Beantwortet Fragen basierend auf FAQ-Daten
- **Ansätze (frei wählbar):**
 - FAQ-Daten im System Prompt
 - Tool Call zum Auslesen der FAQ-Datei
 - String-Match/Keyword-Search
- **Input:** Frage + Kategorie vom Router

Agent 3: Action Agent

- Simuliert Aktionen via Mock-Functions
- **Actions:** "create_ticket", "check_status", "escalate"
- **Output:** Strukturierte Aktionsbestätigung

System-Flow Beispiel

Input: "Meine Rechnung ist falsch, können Sie das prüfen?"

1. Router Agent → Klassifiziert als "billing"
2. Knowledge Agent → FAQ-Lookup für Billing-Fragen

```
3. Action Agent → create_ticket("billing_dispute")
4. Output: "Ticket #123 erstellt, Billing-Team kontaktiert Sie in 24h"
```

Workflow:

```
User Query → Router Agent → Knowledge/Action Agent → Response
```

Was wir bereitstellen

Mock-Daten

```
support_data/
├── faq_data.json (Q&A, nach Kategorien sortiert)
├── customer_data.json (Test-Kunden)
└── test_queries.txt (verschiedene Anfragen)
```

Infrastruktur

- OpenAI API Key (Testbudget)
- Evaluation-Script für Testfälle

Deliverables

1. **Code** (GitHub Repository oder ZIP)
2. **README** mit Setup-Anleitung (max. 1 Seite)
3. **Demo** (CLI mit 3-5 Beispiel-Queries)

Fokus: Agent-Zusammenspiel

Bewertungskriterien:

- **Agent-Kommunikation:** Wie geben Agents Informationen weiter?
- **Workflow-Orchestrierung:** Wer entscheidet welcher Agent als nächstes aktiv wird?
- **Message Passing:** Struktur der Agent-zu-Agent Kommunikation

Core Features

- ☒ **Fokus:** Agent-Koordination und Message-Passing
- ☒ **Einfache Datenstrukturen** (JSON, Dictionaries)
- ☒ **Statische Mock-Daten**
- ☒ **Klare Agent-Rollen und Verantwortlichkeiten**

Der Fokus liegt auf der Architektur und dem Zusammenspiel - nicht auf der Komplexität der einzelnen Agents!

Fragen während der Bearbeitung sind ausdrücklich erwünscht!

