KI-getriebene-Softwareentwicklung-Block-2-Tag-2

Grußinator

🎉 KI-Grußkarten Generator

Ein kreatives Web-Projekt, das auf Basis von Nutzereingaben automatisch eine witzige, emotionale oder stilvolle Grußkarte generiert – bestehend aus einem **Spruch (Text)** und einem **dazu passenden Bild**.

🚀 Vollständig umgesetzt mit:

- HTML/CSS/JavaScript (Frontend)
- Python (FastAPI Backend)
- Für die Spruch generierung wird llama-3.3-70b-versatile verwendet. Die Kommunikation mit dem Modell ist über die qroq API realisiert.
- Zur Bildgenerierung wird das Modell "stable diffusion large" verwendet. Die Kommunikation ist über die Hugginface API realisiert.
- Docker & Docker Compose (Deployment)

🧠 Projektidee

Diese Anwendung erlaubt es Nutzer:innen, **eine Grußkarten-Kategorie** (z. B. Geburtstag, Hochzeit, Geburt, Beerdigung etc.) auszuwählen.
Anschließend werden **kategoriespezifische Felder** angezeigt (Name, Alter, Hobbys etc.), die über ein Formular erfasst werden.

💡 Die Daten werden dann an das **Backend geschickt**, welches:

1. Einen passenden Textspruch generiert
2. Einen Prompt für eine Bild-KI erzeugt
3. Ein fertiges Grußkarten-Bild generiert

Das Ergebnis wird im **Frontend angezeigt**: Text & Bild einer personalisierten Grußkarte ✨

Link zum Repository

https://gitlab.rwu.de/ai-ki-swe/250517-ai-augmented-apps/grusinator

---

Quickstart Guide

Repo klonen

```
git clone https://gitlab.rwu.de/ai-ki-swe/250517-ai-augmented-apps/grusinator
cd grusinator
```

API-Keys konfigurieren

Passe im Verzeichnis backend die `.env` Datei an indem Sie die Passenden API-Keys hier eintragen:

```
HUGGING_FACE_API_KEY=YOUR_API_KEY
GROQ_API_KEY=YOUR_API_KEY
```

Zurück ins Projektverzeichnis

```
cd ..
```

Docker Compose starten

```
docker compose up --build
```

Anwendung aufrufen

```
[http://localhost:80]
```

📁 Projektstruktur

```
.
├── backend/
│   ├── requirements.txt
│   ├── .env
│   ├── request.json
│   └── app/
│       ├── main.py
│       ├── modely.py
│       ├── llm_api.py
│       └── text_to_image
│           ├── description_generation.py
│           └── text_to_image.py
├── frontend/
│   ├── index.html
│   ├── style.css
│   ├── script.js
│   ├── fieldTemplates.js
│   └── images/
│       └── background.png
├── docker-compose.yml
└── README.md
```

🤖 Verwendung von KI-Tools

KI-Tool	Einsatzzweck	Einsatzintensität
ChatGPT-4o	Code-Generierung	Durchgängiger Einsatz zur Code-Generierung im Projekt
GitHub Copilot (Codex)	Code-Vervollständigung	Durchgängiger Einsatz zur Code-Vervollständigung im Projekt