Prototyp
Themenfeld Automatisierung (mit KI)
KI Lern-App für Kommunen

Prototypname
KI Lern-App für Kommunen

Ziel
Beitrag zum Lernen mit und über künstliche Intelligenz im kommunalen Umfeld leisten

Themenfeld / Bedarfscluster
Automatisierung (mit KI) und Tools und Tipps zur Arbeitserleichterung

Ausschreibungsstart / Kickoff
Dezember 2023 / März 2024

Fokus
Micro Learnings als Helfer

Entwicklerfirma
jambit GmbH

Kurzbeschreibung - Funktionalitäten
- KI Informationsplattform mit integrierten Lernelementen, Dashboard, FAQs, Quizze
- Grundinformationen als Micro Learnings für alle kommunalen Mitarbeiterinnen und Mitarbeiter
- Ausbaustufe Version 2 mit OpenAI ChatGPT API

Generelle Prototyp-Anforderungen
- Web-App zur Content-Bereitstellung aus vorhandenen (Web-) Informations- und Datenquellen, sowie neuen erstellten Inhalten
- Web-App im Wesentlichen konzipiert als Informations- und Lernapplikation
- Nutzerzentrierte Benutzerführung
- Bereitstellung der Web-App als Open Source
- Optional: Redaktionssystem zur thematischen Erweiterung

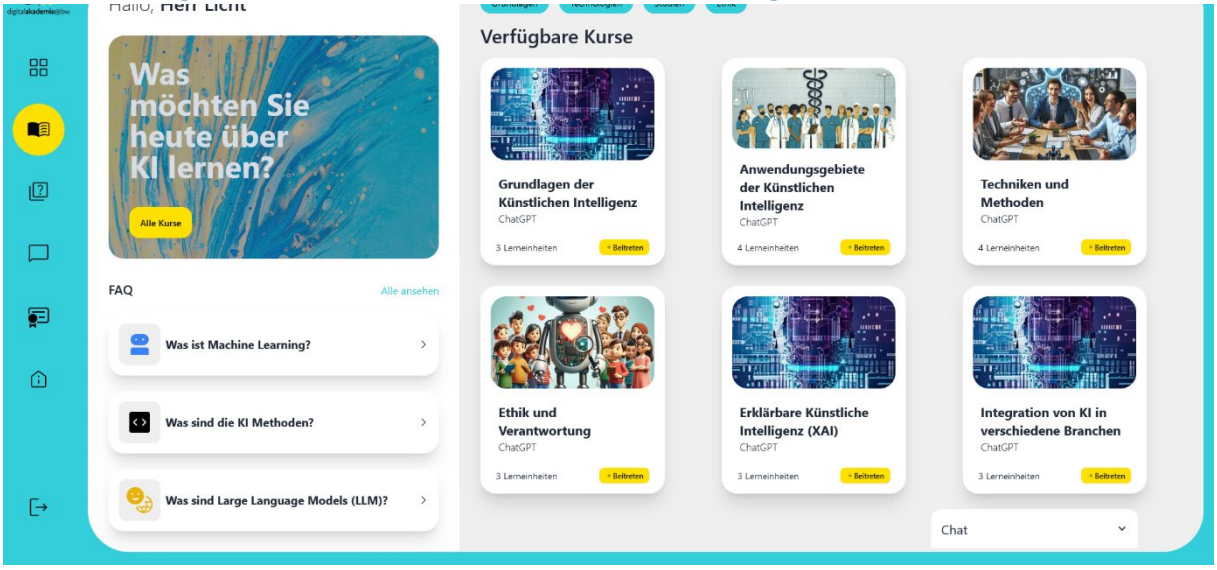
Nutzen
Diese Prototyp Web-App trägt zum Lernen mit künstlicher Intelligenz im kommunalen Umfeld bei. Sie fungiert als erweiterbare Informationsplattform mit integrierten

Lernelementen „Micro Learnings“ und bietet leicht zugängliche KI Grundinformationen für alle kommunalen Mitarbeitenden.

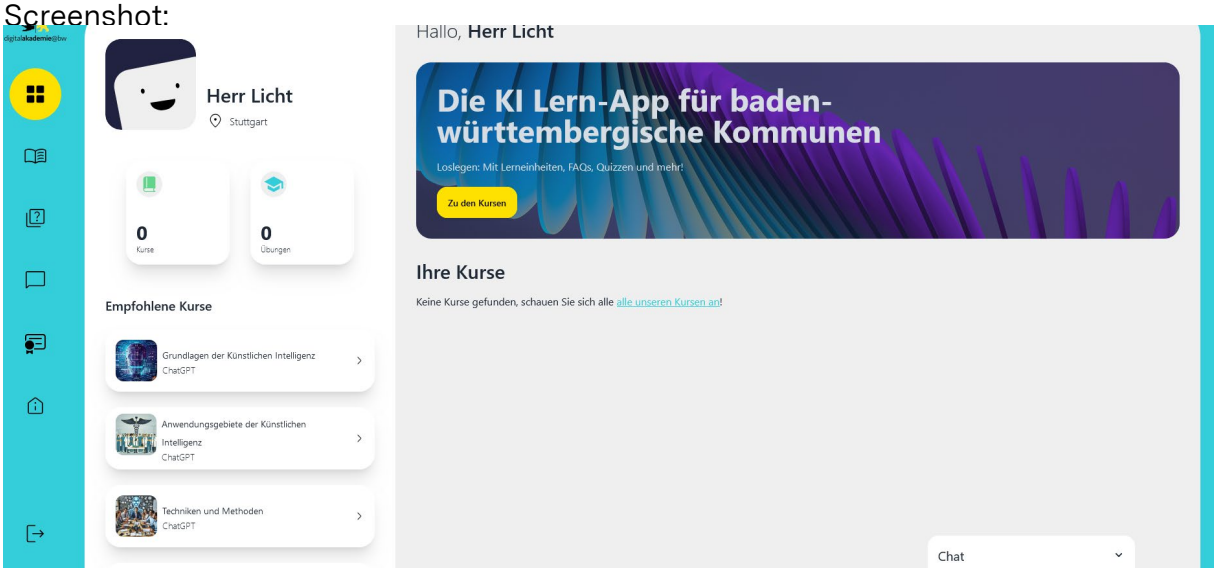
Ausblick 2024

- Einbindung weiterer KI Inhalte
- Erweiterung mit OpenAI ChatGPT API

Darstellungen von Konzeption und Entwicklung



Screenshot:



Kontaktieren Sie uns gerne für Ideen, Feedback und Einbindung in die Entwicklung.
Ihr KommHUB Team.

ai-learning-platform

Project Overview

AI Learning Platform is an interactive and accessible learning platform for artificial intelligence, specifically designed to meet the needs of public sector institutions and municipalities. It provides a low-threshold entry point to AI-related topics through structured learning paths, self-assessment tools, and certification-based testing.

Users can explore curated courses and quizzes to deepen their understanding of key AI concepts. The platform emphasizes awareness of the EU AI Act, offering guidance and sensitization to its implications for public service employees.

Beyond learning content, the platform also includes experimentation tools such as a prompt generator and an integrated AI chat. These allow users to explore prompt engineering hands-on or ask questions directly related to the platform's educational material.

Docker Compose

Included in this project is a simple docker-compose configuration to deploy a built

version of the app with an nginx proxy server.

To ensure functionality, the backend must be publicly accessible via HTTPS so the frontend can reach it over the web.

Additionally, the API key for OpenAI ChatGPT must be provided via the environment variable `API_KEY`. This can be done either through a `.env` file or directly as an environment variable in the deployment environment.

Commands

Prerequisite: Docker and docker-compose are installed and available.

```sh
docker-compose up
```

This command is all that is needed to build and start the server.

The app should be available at `http://localhost:8080` from the machine it is run on.

Frontend

Optional deployment notes

Vue can be built with `npm run build` (assuming `npm install` has previously been run and is up to date).

Building will create a `dist` folder, the contents of which can be hosted on any static

host server. No other services or projects are currently required for the app to function,

including Docker or docker-compose (this configuration was simply included for convenience).

Note: For the Deep Chat integration to work correctly, the endpoint must be set to the publicly accessible HTTPS URL of the backend mentioned above.

Deployment Options

There are multiple ways to deploy this platform, depending on your infrastructure and preference:

1. Docker-based Deployment

Use the provided `docker-compose.yml` to deploy both the frontend and backend as containerized services. This is suitable for:

- Local development and testing

- Small-scale production environments

- Easy deployment to Docker-compatible platforms like **DigitalOcean App Platform**, **Render**, or **Railway**

2. Static Hosting for Frontend + Node Server for Backend

You can host the built Vue frontend (`dist/` folder) on any static file host such as:

- **Vercel**
- **Netlify**
- **GitHub Pages** (if CORS and backend config allow)

The backend (Express server) can be deployed independently on platforms such as:

- **Railway**
- **Heroku**
- **Render**
- **AWS Elastic Beanstalk**
- **Google Cloud Run**
- **VPS providers** like Hetzner or DigitalOcean

Ensure that:

- The backend is served over HTTPS and publicly reachable.
- The endpoint is correctly configured in the frontend to point to the deployed backend.
- The `API_KEY` environment variable is set on the server.

3. Full-stack deployment on platforms like Fly.io or Heroku

You can also bundle both frontend and backend in one Node-based service and deploy them together (e.g. serve the `dist/` folder with Express).

License Information

Used Libraries

* Vue (MIT)
* Vite (MIT)

* vue-I18n (MIT)

* vue-router (MIT)

* Pinia (MIT)

* VueUse (MIT)

* daisyUI (MIT)

* vue-boring-avatars (MIT) / boring-avatars (MIT)

* Iconify for Vue / Iconify (MIT)

* html2canvas (MIT)

* jsPDF (MIT)

* deep-chat (MIT)

* echarts (Apache License V2)

This project uses [Apache ECharts](https://echarts.apache.org/en/index.html), which is licensed under the Apache License 2.0.

See `licenses/apache-2.0-echarts.txt` and `NOTICE-echarts.txt` for details.

Recommended IDE Setup

[VSCode](https://code.visualstudio.com/) +
[Volar](https://marketplace.visualstudio.com/items?itemName=Vue.volar) (and disable Vetur).

Type Support for `.vue` Imports in TS

TypeScript cannot handle type information for `.vue` imports by default, so we replace the `tsc` CLI with `vue-tsc` for type checking. In editors, we need
[Volar](https://marketplace.visualstudio.com/items?itemName=Vue.volar) to make the TypeScript language service aware of `.vue` types.

Customize configuration

See [Vite Configuration Reference](https://vitejs.dev/config/).

Project Setup

```sh
npm install
```

Compile and Hot-Reload for Development

```sh
npm run dev
```

Type-Check, Compile and Minify for Production

```sh
npm run build
```

Run Unit Tests with [Vitest](https://vitest.dev/)

```sh
npm run test:unit
```

Run End-to-End Tests with [Playwright](https://playwright.dev)

```sh
# Install browsers for the first run
npx playwright install

# When testing on CI, must build the project first
npm run build
```

```
# Runs the end-to-end tests
npm run test:e2e
# Runs the tests only on Chromium
npm run test:e2e -- --project=chromium
# Runs the tests of a specific file
npm run test:e2e -- tests/example.spec.ts
# Runs the tests in debug mode
npm run test:e2e -- --debug
```

Lint with [ESLint](https://eslint.org/)

```sh
npm run lint
```