1. Prerequisites

Before installing the *HARA-Assistant* , make sure you have **Python 3.11.x** installed on your system. The project was developed using **Python 3.11.9**, so this version is recommended for maximum compatibility. Other patch versions within the 3.11 series (such as 3.11.8 or 3.11.10) may also work, but they have not been explicitly tested.

You can check your python version in Windows by opening the terminal/cmd and running:

```
...>python --version
```

or for Linux/macOS:

```
...>python3 --version
```

If Python is not installed or if you're using a different version, follow the instructions at https://www.python.org/downloads/release/python-3119/ to download and install Python 3.11.9.

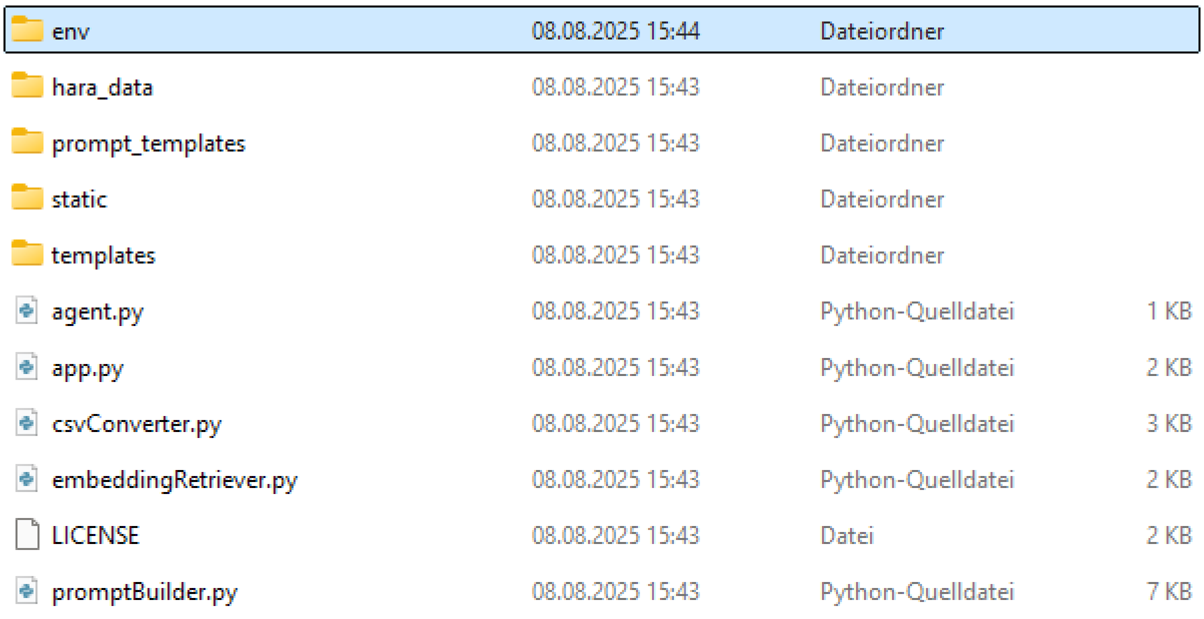
2. Set Up a Virtual Environment (Recommended)

It is strongly recommended to create a **virtual environment** for the HARA-Assistant This ensures that all dependencies are installed in an isolated environment and will not interfere with other Python projects or global packages on your system. To set up a virtual environment navigate to the folder where you have extracted the repository of the HARA-Assistant open the terminal again and type the following command:

```
...>python -m venv env
```

This will set up a virtual environment with the name **env**, but you can give it any other name you want, just be aware that your chosen name should then be displayed instead of **env** for the following sections.

After running the command it should look like the following:



env	08.08.2025 15:44	Dateiordner	
hara_data	08.08.2025 15:43	Dateiordner	
prompt_templates	08.08.2025 15:43	Dateiordner	
static	08.08.2025 15:43	Dateiordner	
templates	08.08.2025 15:43	Dateiordner	
agent.py	08.08.2025 15:43	Python-Quelldatei	1 KB
app.py	08.08.2025 15:43	Python-Quelldatei	2 KB
csvConverter.py	08.08.2025 15:43	Python-Quelldatei	3 KB
embeddingRetriever.py	08.08.2025 15:43	Python-Quelldatei	2 KB
LICENSE	08.08.2025 15:43	Datei	2 KB
promptBuilder.py	08.08.2025 15:43	Python-Quelldatei	7 KB

After this we can activate the virtual environment via the terminal with the following command

```
...>env\Scripts\Activate
```

or for Linux/macOS:

```
...>source env/bin/activate
```

now you should get a `(env)` text in front of your terminal line that indicates the virtual environment is active:

```
(env) ...>
```

FYI: once you **don't need the environment anymore** you can deactivate the virtual environment via:

```
(env) ...>deactivate
```

You can check if this worked, if you don't see the `(env)` anymore in front of the terminal line. To reactivate the virtual environment simply use env\Scripts\Activate again

3. Installing Dependencies

Now that the virtual environment is set up and activated, you should see a `(env)` or similar indicator at the beginning of your command line. This confirms that you're working inside the isolated environment. With that in place, we can now check if the virtual environment is fresh via:

```
(env)...>pip list
```

Which should display something like the following, with a minimal amount of packages installed.

```
Package    Version

---------- -------

pip        24.0

setuptools 65.5.0
```

You can check the difference by deactivating the virtual environment and running the command again, which will probably display a lot more packages in your global environment. Make sure you activate the virtual environment again if you try this. Additionally when running this command after you installed all the dependencies it should display them.

First we install the dependencies related to the Open-router which we need to call the Llama 3.3 70B instruct model, which uses the OpenAi framework. For this we type

```
(env)...>pip install openai
```

Next we take care of the dependencies related to flask which takes care of our backend, enabling the python code to talk to the javaScript code in the frontend via:

```
(env)...>pip install flask
```

Next we take care of the dependencies for the .env file we will create later in this guide:

```
(env)...>pip install python-dotenv
```

Finally we will take care of all the dependencies related RAG-pipeline.

```
(env)...>pip install langchain
```

and

```
(env)...>pip install langchain-ollama
```

and

```
(env)...>pip install langchain-chroma
```

and

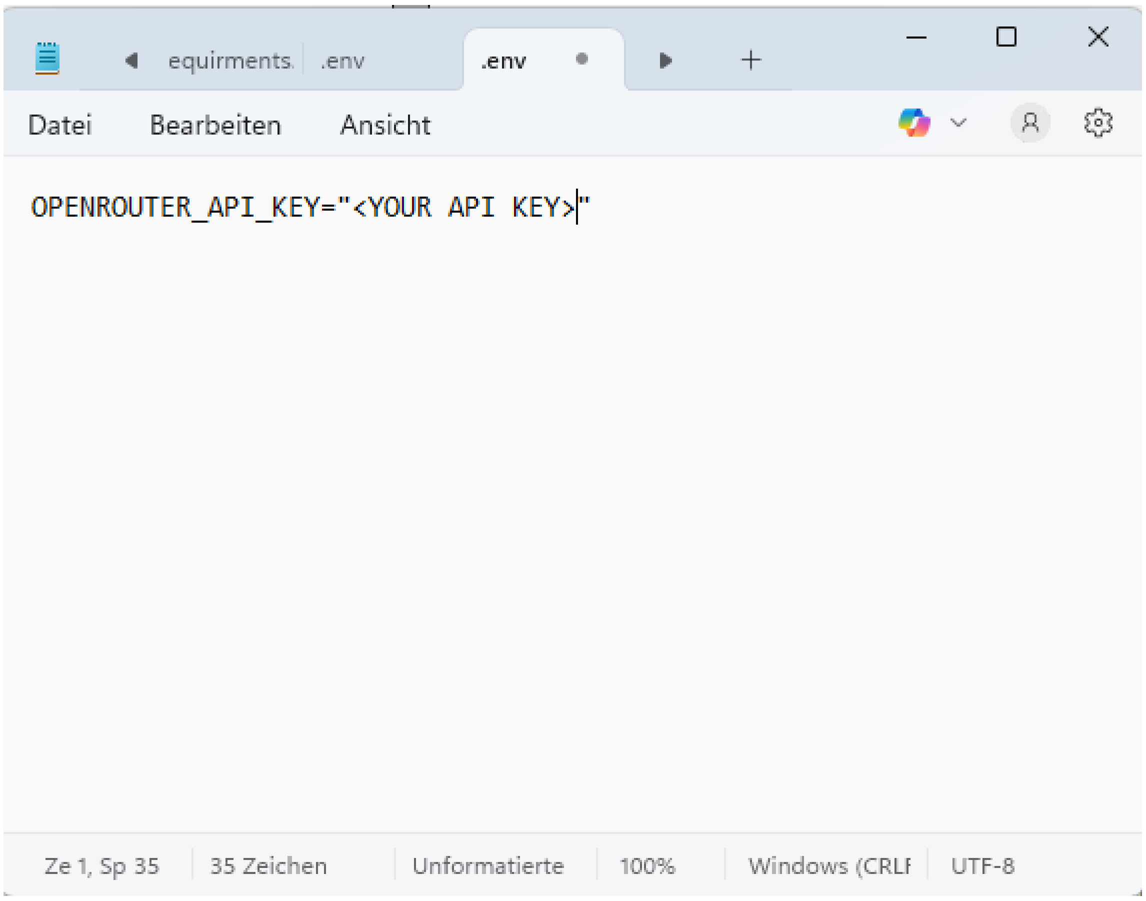
```
(env)...>pip install pandas
```

After all of these are successfully installed, the last two things we need to take care of is installing Ollama for the embedding model and getting an API key for Gemini 2.5 Flash.

4. API-KEY and Ollama

First lets take care of the API key that is needed to a send requests to Open-router which we need to call the Llama 3.3 70B instruct model . To get one open the following link: https://openrouter.ai/settings/keys

Once you have you key, the next task is to create a file named ".env" on the top layer of the Free-ISO-26262-chatbot folder where all the python files also are . In this file you write the following, where you exchange <YOUR API KEY> with the string that is your API key:



```
OPENROUTER_API_KEY="<YOUR API KEY>"
```

That takes care of the Open-router side of things and the code can now make an API call to Llama 3.3 70B instruct model.

It is important to note at this point that the code uses the Llama 3.3 70B instruct (free) model. Meaning this model is rate limited to 50 requests per day. If you purchase credits worth 10€ at the side, your rate limit will increase to 1000 calls for free models a day permanently.

Next we need to download Ollama. You can download the software here https://ollama.com/download . Ollama is a tool that lets you run open source LLMs locally. In our case we need a embedding model, namely the `mxbai-embed-large`. Once Ollama is installed and running, you can check it if you open your terminal again an type :

```
...>ollama
```

This should show you some helpful usage tips and also confirm that the software is installed and running. (FYI: You don't need your virtual environment (env) running at this point)

Next we need to pull the “mxbai-embed-large”. For this we type

```
...>ollama pull mxbai-embed-large
```

You can check of it is installed by typing:

```
...>ollama list
```

Once that has finished we are all set up. It is important to note here tho that the `mxbai-embed-large` is an embedding model tjat will run locally on your computer and thereby would greatly benefit from the presence of a GPU. The documentation states that CPU mode is also supported, but this hasn't been tested by the author, so if one runs into issues when first running the HARA-Assistant app, where the vector store is build, that is most likely the problem.

5. Start-up the HARA-Assistant

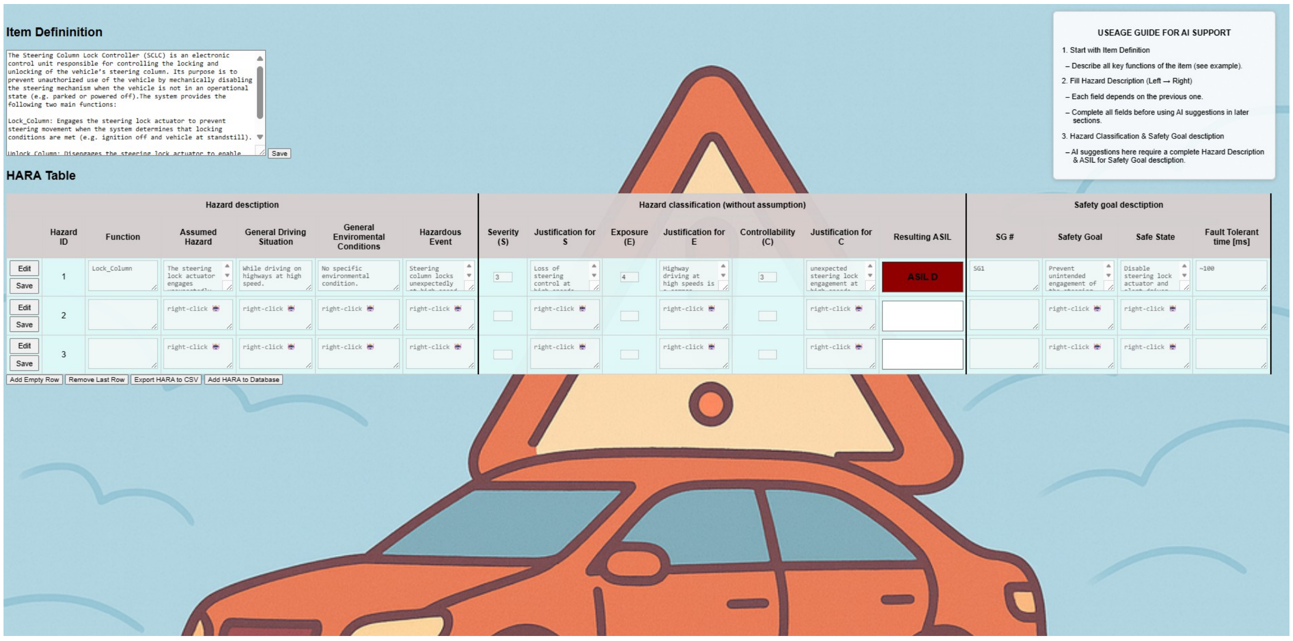
Now we can finally start the app. For this we navigate into the HARA-Assistant folder where the app.py file is located. Then we activate the virtual environment as before (if it is not already running). Once the virtual environment is running we can type

```
(env)...>python app.py
```

This should now start the HARA-Assistant. When you first start the HARA-Assistant it will most likely take a little bit of time until it starts as it needs to build the vector store. This will occur only the first time (as long as you don't delete the “chrome_langchain_db” folder). You will also be informed that this process may take a while in your terminal. Once its finished a new tab should open in your browser which looks as follows:

The usage is relativly simple, type your question in the field and press send.

To quite the chatbot there are two options, ether you press “ctrl+c” in your terminal or you close the terminal where you started the app. Both will shut down the app.



Item Defininition

```
The Steering Column Lock Controller (SCLC) is an electronic
control unit responsible for controlling the locking and
unlocking of the vehicle's steering column. Its purpose is to
prevent unauthorized use of the vehicle by mechanically disabling
the steering mechanism when the vehicle is not in an operational
state (e.g. parked or powered off).The system provides the
following two main functions:

Lock_Column: Engages the steering lock actuator to prevent
steering movement when the system determines that locking
conditions are met (e.g. ignition off and vehicle at standstill).
```

USEAGE GUIDE FOR AI SUPPORT

1. Start with Item Definition
   – Describe all key functions of the item (see example).
2. Fill Hazard Description (Left → Right)
   – Each field depends on the previous one.
   – Complete all fields before using AI suggestions in later sections.
3. Hazard Classification & Safety Goal desctiption
   – AI suggestions here require a complete Hazard Description & ASIL for Safety Goal desctiption.

HARA Table

	Hazard desctiption						Hazard classification (without assumption)							Safety goal desctiption			
Hazard ID	Function	Assumed Hazard	General Driving Situation	General Enviromental Conditions	Hazardous Event	Severity (S)	Justification for S	Exposure (E)	Justification for E	Controllability (C)	Justification for C	Resulting ASIL	SG #	Safety Goal	Safe State	Fault Tolerant time [ms]
1	Lock_Column	The steering lock actuator engages	While driving on highways at high speed.	No specific environmental condition.	Steering column locks unexpectedly	3	Loss of steering control at	4	Highway driving at high speeds is	3	unexpected steering lock engagement at	ASIL D	SG1	Prevent unintended engagement of	Disable steering lock actuator and	~100
2		right-click	right-click	right-click	right-click		right-click		right-click		right-click			right-click	right-click	
3		right-click	right-click	right-click	right-click		right-click		right-click		right-click			right-click	right-click	

6. Customization options

This section briefly describes some customization options your have. If you open the folder “prompt_templates”, are various system prompt templates and user prompt templates. The system prompt (__<field name>_system) templates guide the behavior of the LLM for this field , while the user prompt (__<field name>_user) handles the input information that is given to the LLM for generation. If you observe some unwanted behavior if the LLM or want to customize it's behavior to your needs, system prompts are the place where you can change things around and see how they play out. Giving additional information with the use of user prompt can also be done, just follow the convention, but it is not recommended for the Llama 3.3 70B instruct. Only if you decide to replace the LLM with a different one (e.g. Googles Gemini) this might be a viable option.

Additionally in the “embeddingRetriever.py” file one can play around a bit with the retrieving mechanism. The algorithm used and how many text chunks it retrieves is defined on the bottom as

```
retriever = vector_store.as_retriever(...)
```

There are also other options besides the one used if one want to read up these. The number of retrieved examples can also be increased, but for those to have an effect also follow the convention in the system prompts (e.g. __assumed_hazard_system.txt) where retrieved examples are used. When using a different (more powerful LLM) using retrieved examples might also be beneficial for fields that currently use no retrieval augmentation.

7. Usage Guide

There exist several buttons in the UI as one can see. “Add Empty Row” and “Remove Last Row” should be relatively self explanatory.

There also exists the option to export the data to a CSV file as well as add the current HARA table to the HARA database so it can be retrieved as a potential example for future runs. It is **important** to know that a HARA row will only be exported if it is complete. Meaning if any fields are left empty that row will be ignored for the export. If one does not want to fill out a field e.g. because it is QM and does not need Safe State options, one should ether use a “-” or better “QM does not need <field_name>”.

If one exports to augment the HARA database with the examples, as already mentioned only complete rows will be exported plus, to make use of the expanded HARA database, one must delete the “chroma_langchain_db” folder so the program builds a new database, otherwise the old will still be used.

FYI: if the app gets an error while starting related to pandas, most likely the HARA_database.csv file is for some reason not correctly formated. If you get such an error make first of all sure that the correct amount of entries are in each row and second that there is an empty row at the end. Otherwise the problem occurs where the first row of the contend to be added will be added to the last present row. This problem should only occur if you have for some reason edited the HARA database manually.