1. Prerequisites

Before installing the *Free-ISO-26262-Chatbot*, 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.

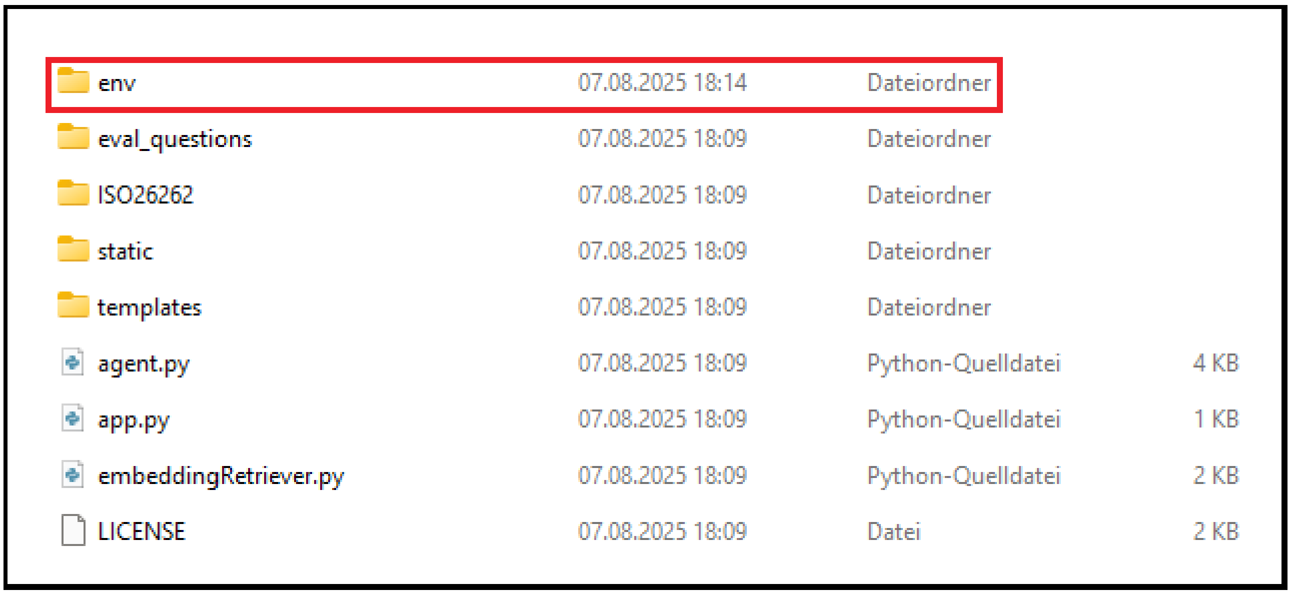
2. Set Up a Virtual Environment (Recommended)

It is strongly recommended to create a **virtual environment** for the Free-ISO-26262-Chatbot. 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 enviroment navigate to the folder where you have extracted the repository of the Free-ISO-26262-Chatbot 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	07.08.2025 18:14	Dateiordner	
eval_questions	07.08.2025 18:09	Dateiordner	
ISO26262	07.08.2025 18:09	Dateiordner	
static	07.08.2025 18:09	Dateiordner	
templates	07.08.2025 18:09	Dateiordner	
agent.py	07.08.2025 18:09	Python-Quelldatei	4 KB
app.py	07.08.2025 18:09	Python-Quelldatei	1 KB
embeddingRetriever.py	07.08.2025 18:09	Python-Quelldatei	2 KB
LICENSE	07.08.2025 18:09	Datei	2 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) ...>
```

once you are finished **with the entire guide** you can deactivate the virtual environment via:

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

You can check if this worked, if you dont see the `(env)` anymore in front of the terminal line. To reactivate the virtual enviroment 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 enviroment and running the command again, which will propably display a lot more packages in your global enviroment. 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 diesplay them.

First we install the dependencies related to the google Gemini 2.5 Flash we need to call. For this we type

```
(env)...>pip install google-genai
```

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
```

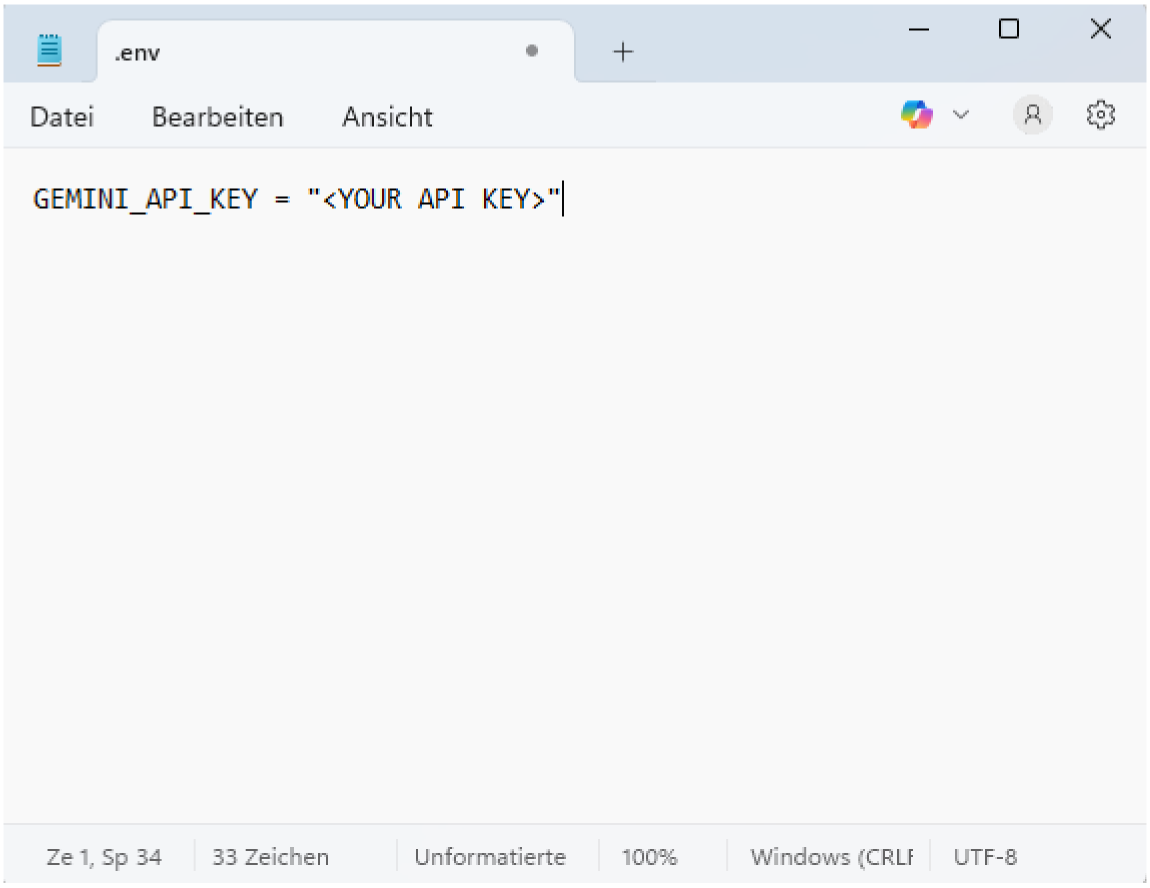
After all of these are successfully installed, the last two things we need to take care of is installing Ollama for the embedding model and getting an API key for Gemini 2.5 Flash.

4. API-KEY and Ollama

First lets take care of the API key that is needed to a send requests to Gemini 2.5 Flash. To get one open the following link: https://ai.google.dev/gemini-api/docs

From there you can press the “Get API Key” button and on the following side on “+ Create API key”. Make sure you copy the generated API key somewhere save where you find it again, because after you closed that window you wont be able to see it again. If you didn't copy it or lost it you should delete the key and request a new one.

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:



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

That takes care of the google side of things and the code can now make an API call to Gemini 2.5 Flash.

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
```

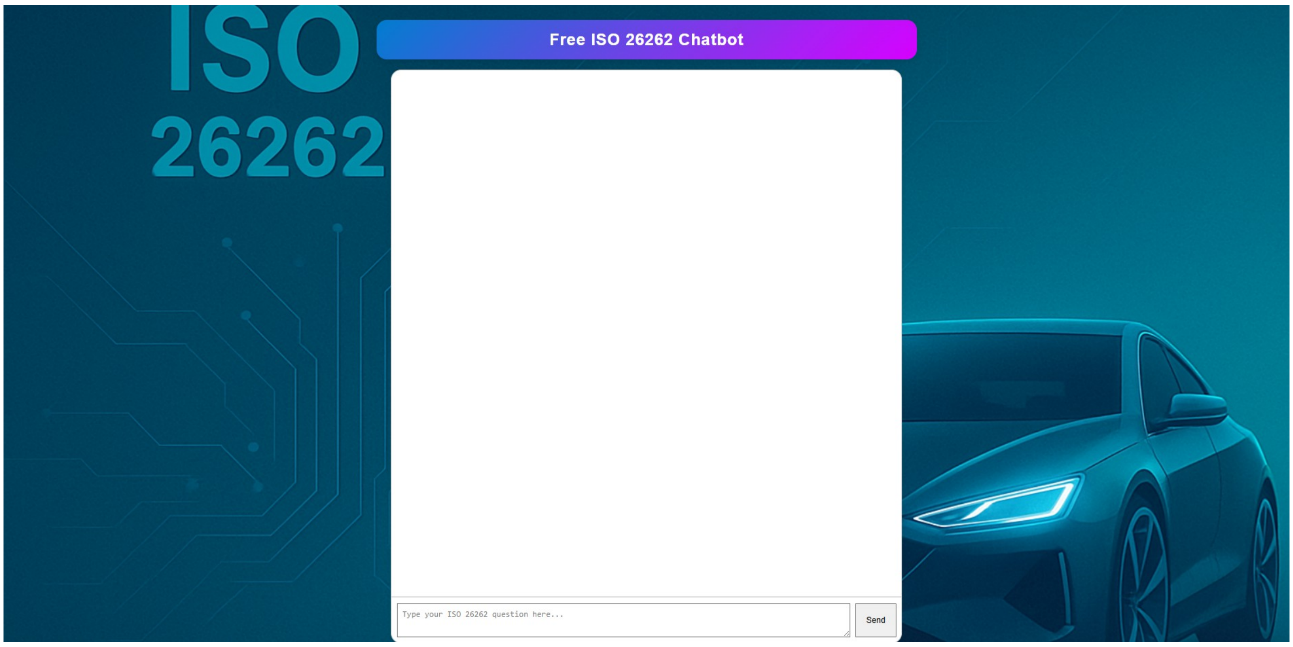
Once that has finished we are all set up. It is important to note here tho that the `mxbai-embed-large` is an embedding model tjat will run locally on your computer and thereby would greatly benefit from the presence of a GPU. The documentation states that CPU mode is also supported, but this hasn't been tested by the author, so if one runs into issues when first running the Free-ISO-26262-Chatbot app, where the vector store is build, that is most likely the problem.

5. Start-up the Free-ISO-26262-Chatbot

Now we can finally start the chatbot. For this we navigate into the Free-ISO-26262-Chatbot 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

```
...>python app.py
```

This should now start the chatbot. When you first start the Free-ISO-26262-Chatbot it will most likely take a few minutes until it starts as it needs to build the vector store. This will occur only the first time (as long as you dont delet 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.

6. Customization options

This section briefly describes some customization options your have. If you open the file “agent.py”, there is a string at the beginning called `system_prompt` which defines the behavior of the Gemini 2.5 Flash. If you observe some unwanted behavior if the LLM or want to customize it's behavior to your needs, here you can change things around and see how they play out.

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(...)
```

with some other options commented out. There are also other options besides these. It is important to note that if one changes for example the number chunks retrieved it might be beneficial to adjust the number in the system prompt since the LLM is made aware of the exact number of text chunks it will receive.