Documentation: LeuphAI Q&A

tech basics final project Lennox Krause

Initial idea

This is the initial idea I had on what to create and which steps to take creating it:

- A useful AI assistant for students studying at Leuphana
- Using a preexisting AI model and implementing it into python
- Finetuning its answers and training it
  - Maybe using Leuphana website
  - Maybe even creating a website crawler, looked quite complicated, but maybe it could be possible
- Maybe implementing it into some kind of interface if it's not enough coding by then

Initial code structure for idea

This is the structure I had in my head:

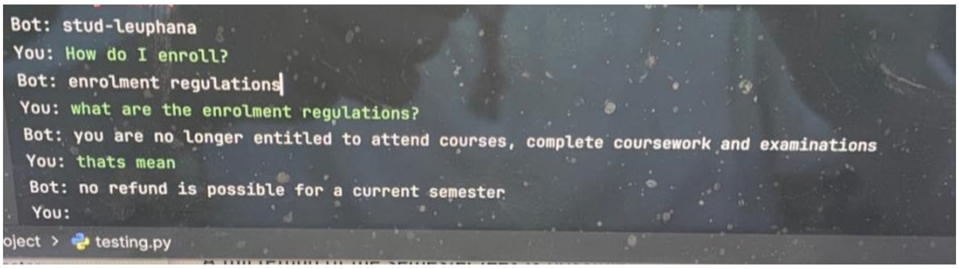
- While loop
- Player inputs and bot answers
- Possibility to exit
- Code for finding the right context
  - Website crawler
  - Context file?
- Pygame or TKinter interface for better graphics and more coding?
- Fine tuning stuff (I still need to learn details)

About
LEUPHANA
HELPER
Close

Leuphana Helper: Hi, how can I assist you today?
You: When does the semester start?
Leuphana Helper: It starts on the 13th of October
You: How do I enrol
Leuphana Helper: You can enrol via this link: ...
You need...

Initial Research:

- Asked ChatGPT and created a small demo version with its help (1)



```
Bot: stud-leuphana
You: How do I enroll?
Bot: enrolment regulations
You: what are the enrolment regulations?
Bot: you are no longer entitled to attend courses, complete coursework and examinations
You: thats mean
Bot: no refund is possible for a current semester
You:
oject > testing.py
```

This is what the protocol looks like:

Me: Do you think I could impement some form of LLM into python?

ChatGPT: Absolutely — you can implement a lightweight form of an LLM (large language model) in your Python chatbot project, especially using prebuilt APIs or small local models, depending on your setup and time constraints.

Here are a few options, ranked by complexity and feasibility for a 1-month final project:

✅ Option 1: Use OpenAI's GPT API (Highly Feasible)

Best for: impressive results with little setup

What you can do:

Connect your Python chatbot to GPT-3.5/4 via the OpenAI API

Send the user’s message as a prompt and return the response

Add your own logic around it (e.g., fallback replies, custom personality)

Pros:

Very powerful output

Easy to integrate using openai Python library

Looks great in a presentation

Cons:

Requires an internet connection and an API key

Free usage is limited unless you're on OpenAI Plus

🧠 Option 2: Use a Local LLM (Challenging but doable)

Best for: showcasing understanding of models and offline capability

Tools:

llama-cpp-python to run small versions of LLaMA or Mistral

transformers + AutoModelForCausalLM (for HuggingFace models)

Pros:

No need for internet/API keys

Great for learning how inference works

Cons:

Might be slow or limited on a regular laptop

Requires installing and managing large model files (~1–4 GB)

🧪 Option 3: Use a Tiny ML Model for Intent Classification

> Best for: more custom logic, not general chat
>
> You can train a simple model using scikit-learn or transformers to classify user intent (e.g., weather query, joke request, greeting), and route accordingly.
>
> 🧑🏫 Presentation Tips
>
> If you use an LLM, you can:
>
> Compare it with your rule-based logic ("see how the response quality improves")
>
> Talk about safety/guardrails you added (e.g., filtering certain words)
>
> Highlight how you integrated it ("used OpenAI API with custom fallback system")
>
> 🏁 Recommendation
>
> For your final project, I strongly recommend:
>
> Using OpenAI API (or Anthropic, Gemini, etc.) for power + simplicity
>
> Adding your own "shell" of logic around it — so you're not just calling an API, but crafting a product

- Got more resources for learning AI implementation in python from both ChatGPT and our seminary teacher
- Wanted to find a data security conform AI system
    - Found EuroLLM, but got the feedback that it would not work very good for this task

Development process

Problem: The Output is very bad

- How can I improve the output of my AI?
    - Better context data
    - Algorithms for processing context data
    - Better suited AI model
- First idea: improving the output with better context data -> WebCrawler and also some processing attempts
    - Found a tutorial on creating a WebCrawler on the OpenAI website (2)
    - After many hours, the WebCrawler I had created pulled a lot of text from the website, but the processing did not work out quite right
    - I tried out a few different ways to process the data, from the OpenAI website and elsewhere, but either the context was too

        - processed with no data left, or too big with too much data for my laptop to handle (took 16 GB RAM!)
        - So, after wasting a lot of time, I chose to copy the data myself, and to only focus on one aspect of the website (Leuphana music centre)
        - After a bit of trial and error, I restructured the context from the website manually to make it even more readable
    - Second idea: better suited AI model
        - First, I exchanged the EuroLLM model with Roberta, a model from Germany
        - Then, I got the idea to mix Q&A answer finding with text generation to create better suited texts
        - This may have increased the time it takes until the answer is generated, but it also immensely improved the quality of the answers
    - Creating a UI
        - I also wanted to implement some TKinter to do some more actual coding, as the goal of this final project task is also to showcase the knowledge on python I gained attending this course
        - Mainly used one website explaining almost everything needed to understand TKinter (3)

Future improvements and known limitations

- Biggest known limitation: only focusing on Leuphana music centre
    - Two solutions: Either increasing the context and using a device with more computing power or implementing some kind of live WebCrawler which finds context for the answer while its being asked
- Implementing more UI options
    - A “help”/about button giving background information about the chatbot
    - Different tabs for different conversations
- HTML embedding for using it on the Leuphana website

Most important references

(1)www.chatgpt.com (initial AI code)
(2) https://platform.openai.com/docs/tutorials/web-qa-embeddings
(3) https://www.pythontutorial.net/tkinter/tkinter-hello-world/
https://huggingface.co/learn/llm-course/chapter1/1