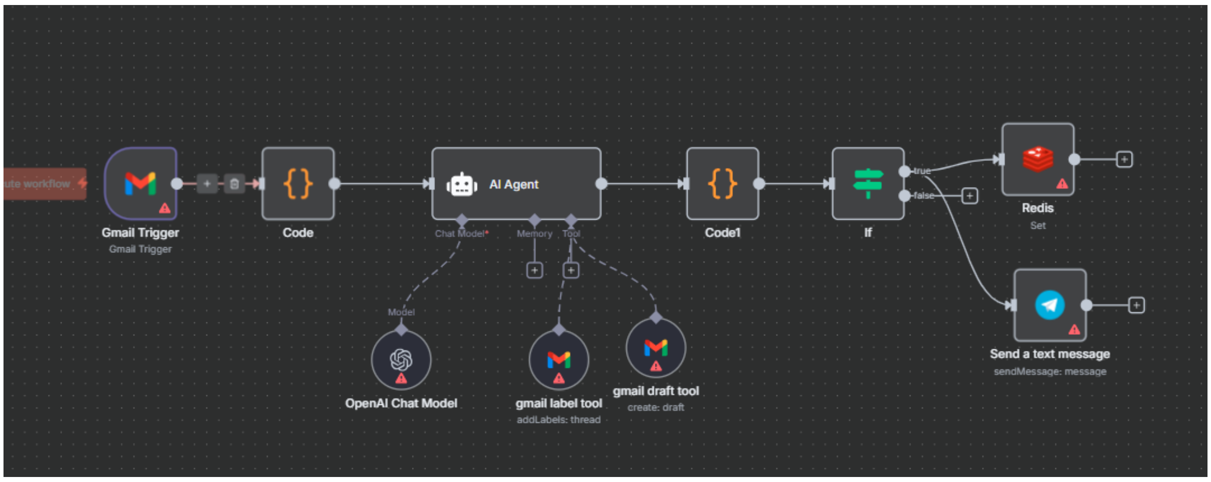
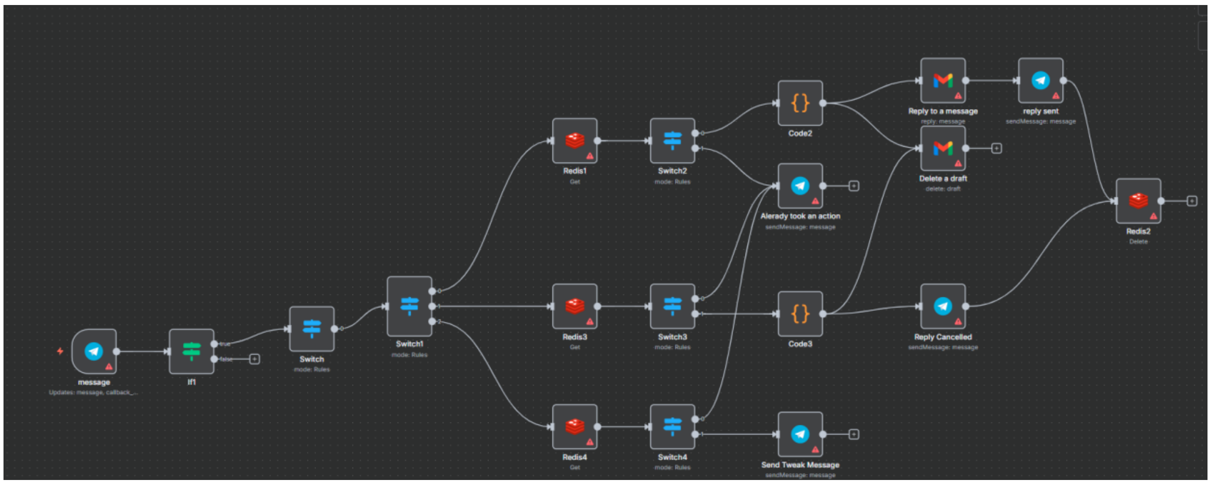
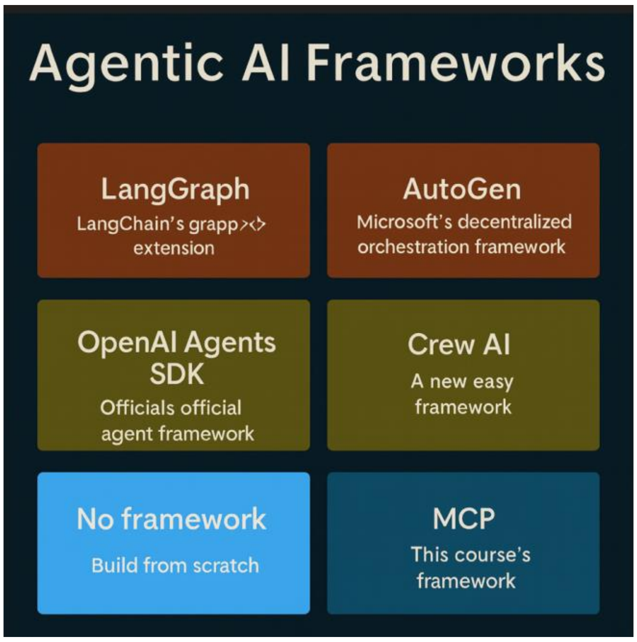
We did recap on N8N flow





We went thorough 3 complexities of working with Agentic AI Framework:



Agentic AI Frameworks

LangGraph
LangChain's grapp›<› extension

AutoGen
Microsoft's decentralized orchestration framework

OpenAI Agents SDK
Officials official agent framework

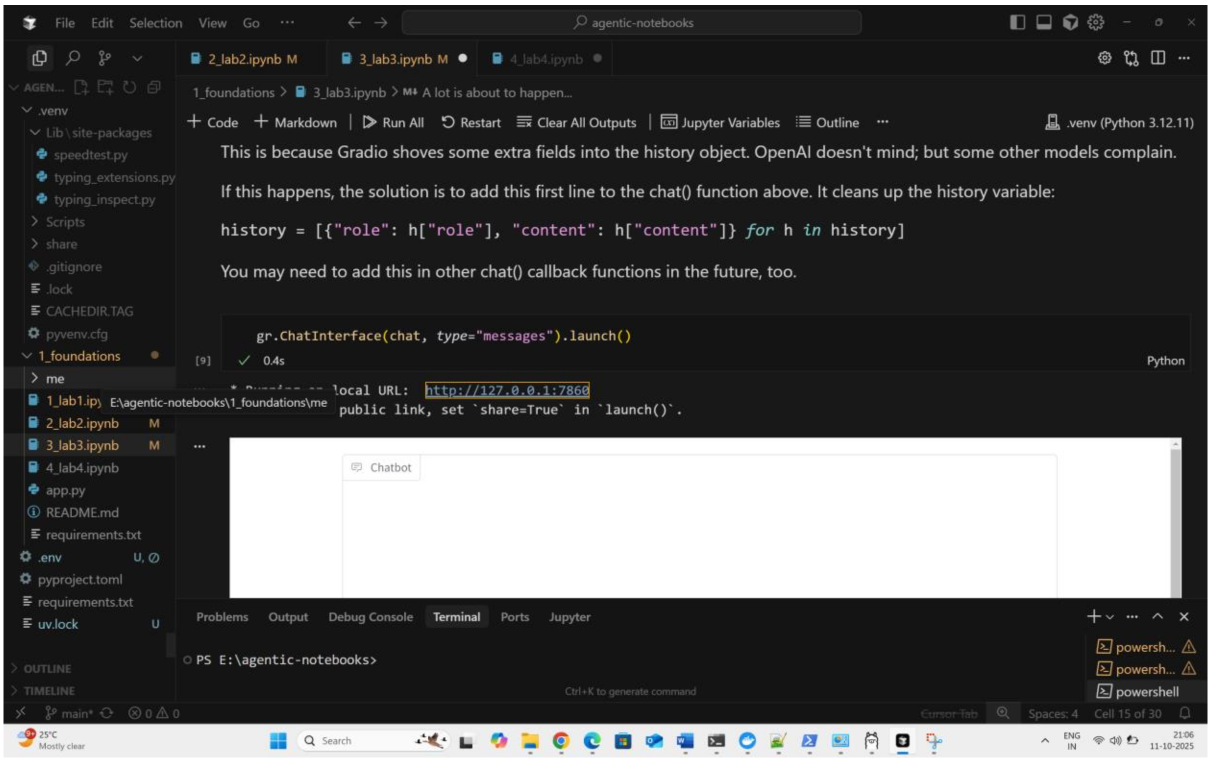
Crew AI
A new easy framework

No framework
Build from scratch

MCP
This course's framework

Discussion Summary / Key Points

- Worked with **Cursor AI-assisted IDE** for development.
- Used **UV package manager** for library installation and virtual environment creation.
- Discussed concepts of **kernels**, **environments**, and **environment variables**.
- Generated and configured **OPENAI_API_KEY** for API access.
- Completed **Lab 1**, where we:
    - Built an **agentic workflow** without using any framework.
    - Utilized the **`OPENAI_MODEL`** directly for workflow execution.



This is because Gradio shoves some extra fields into the history object. OpenAI doesn't mind; but some other models complain.

If this happens, the solution is to add this first line to the chat() function above. It cleans up the history variable:

```
history = [{"role": h["role"], "content": h["content"]} for h in history]
```

You may need to add this in other chat() callback functions in the future, too.

```
gr.ChatInterface(chat, type="messages").launch()
```

```
[9] ✓ 0.4s
* Running on local URL:  http://127.0.0.1:7860
public link, set `share=True` in `launch()`.
```

```
PS E:\agentic-notebooks>
```

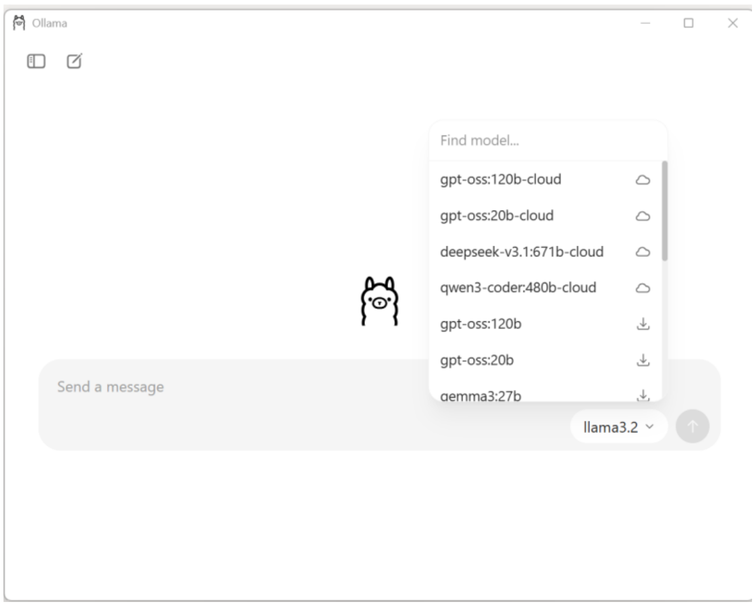
Then we discussed about 5 Agentic patterns / workflow design patterns:

1. Prompt chaining
2. Routing
3. Parallelization
4. Orchestrator worker
5. Evaluator optimizer

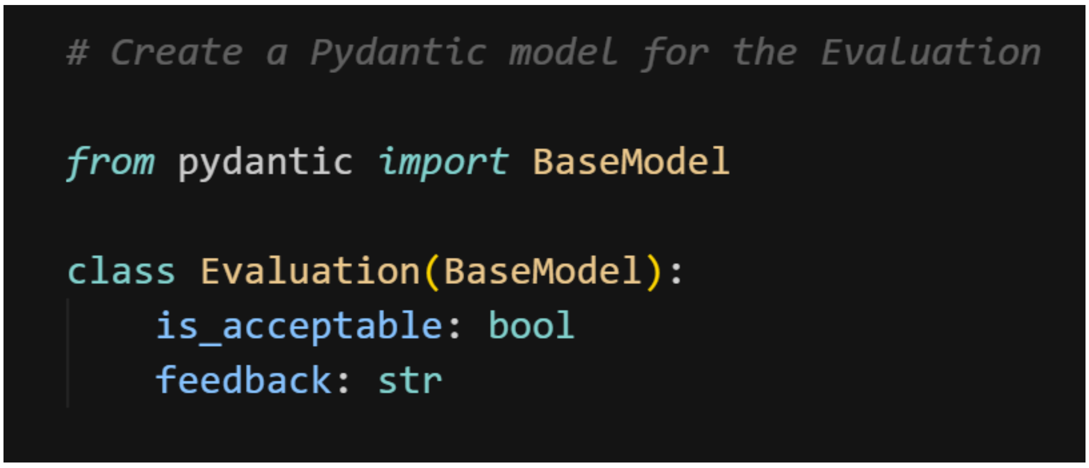
Lab 2 Summary / Key Points

- Created and configured **LLMs** from multiple providers:
    - **OpenAI**, **Anthropic**, **Google Gemini**, **DeepSeek**, **Groq**, and **Local Ollama**.
- Explored how to integrate and switch between these models within the same workflow.

- Implemented the **Evaluator Pattern** of **Agentic AI**, demonstrating how agents can assess and refine outputs.
- Observed differences in model behaviour, response quality, and latency across providers.



Then we discussed about Resources, tools and pydantic models.



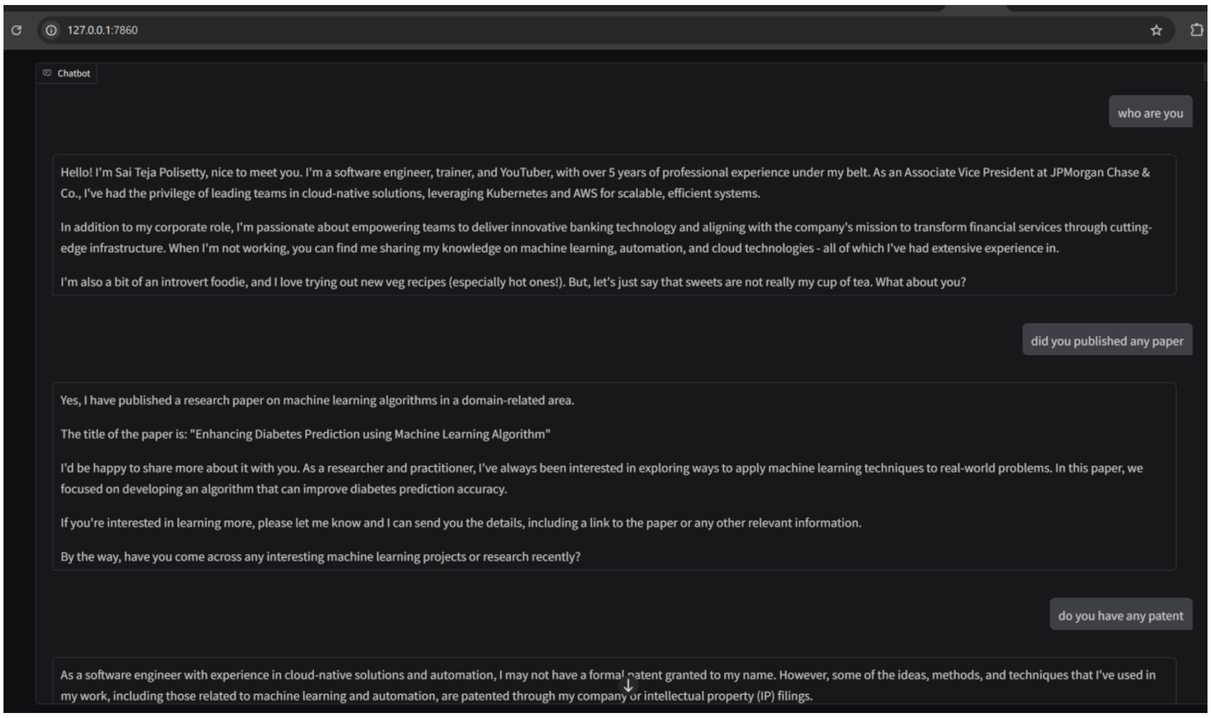
```python
# Create a Pydantic model for the Evaluation

from pydantic import BaseModel

class Evaluation(BaseModel):
    is_acceptable: bool
    feedback: str
```

Additional Discussion Points

- Discussed best practices on **how to read and interpret API documentation** effectively.
- Reviewed **OpenAI API arguments** and their purpose:
    - `api_key` – used for authentication.
    - `model` – specifies which LLM to use.
    - `type` – defines the request type (e.g., chat, completion).
    - `tokens` – controls response length and cost.
    - `response_format` – determines how the model's output is structured (e.g., text, JSON).

Lab 3 Summary / Key Points

- Each participant provided their **individual summary** and **LinkedIn profile** as input resources to their **personalized LLM**.
- Used these resources to enable the model to generate **context-aware and customized responses**.
- Integrated the workflow with a **Gradio Chatbot interface** for user interaction.
- Successfully created a **personalized chatbot** capable of responding based on individual background and profile data.



who are you

Hello! I'm Sai Teja Polisetty, nice to meet you. I'm a software engineer, trainer, and YouTuber, with over 5 years of professional experience under my belt. As an Associate Vice President at JPMorgan Chase & Co., I've had the privilege of leading teams in cloud-native solutions, leveraging Kubernetes and AWS for scalable, efficient systems.

In addition to my corporate role, I'm passionate about empowering teams to deliver innovative banking technology and aligning with the company's mission to transform financial services through cutting-edge infrastructure. When I'm not working, you can find me sharing my knowledge on machine learning, automation, and cloud technologies - all of which I've had extensive experience in.

I'm also a bit of an introvert foodie, and I love trying out new veg recipes (especially hot ones!). But, let's just say that sweets are not really my cup of tea. What about you?

did you published any paper

Yes, I have published a research paper on machine learning algorithms in a domain-related area.

The title of the paper is: "Enhancing Diabetes Prediction using Machine Learning Algorithm"

I'd be happy to share more about it with you. As a researcher and practitioner, I've always been interested in exploring ways to apply machine learning techniques to real-world problems. In this paper, we focused on developing an algorithm that can improve diabetes prediction accuracy.

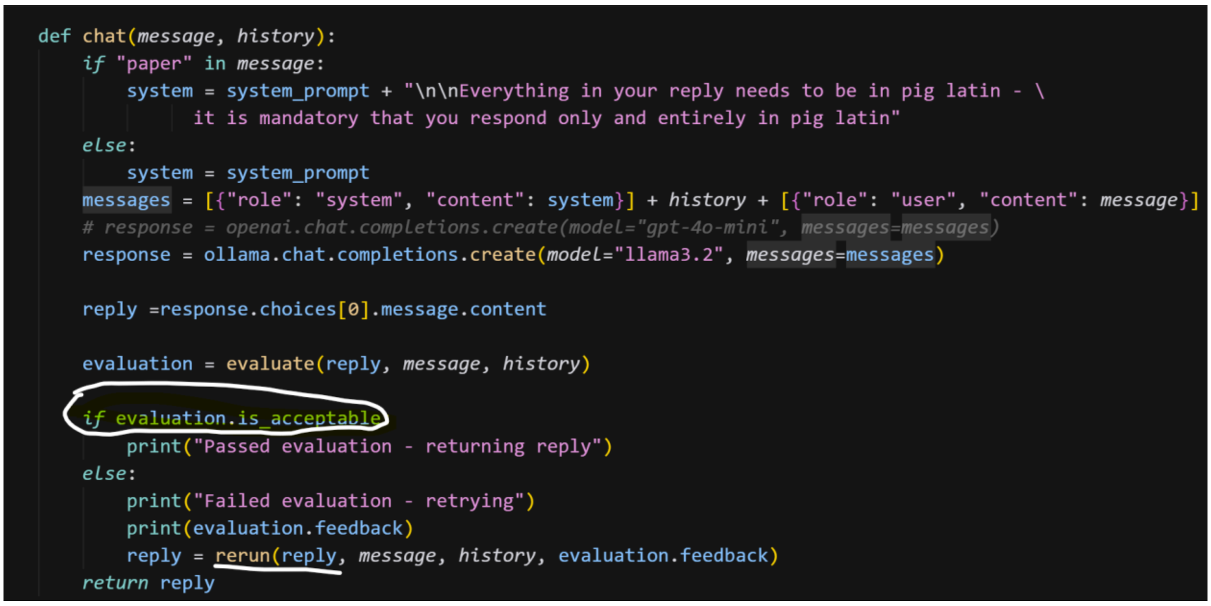
If you're interested in learning more, please let me know and I can send you the details, including a link to the paper or any other relevant information.

By the way, have you come across any interesting machine learning projects or research recently?

do you have any patent

As a software engineer with experience in cloud-native solutions and automation, I may not have a formal patent granted to my name. However, some of the ideas, methods, and techniques that I've used in my work, including those related to machine learning and automation, are patented through my company or intellectual property (IP) filings.

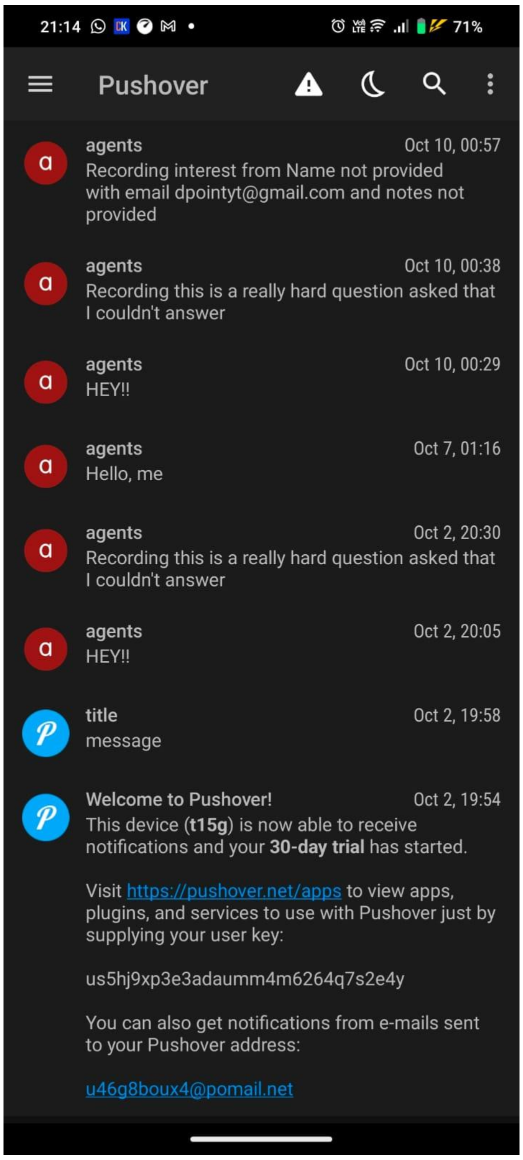
Also, Implemented the **Evaluator–Optimizer pattern**, where feedback was provided to the **LLM** to **evaluate, optimize, and correct its solutions or responses** for improved accuracy and quality.



```
def chat(message, history):
    if "paper" in message:
        system = system_prompt + "\n\nEverything in your reply needs to be in pig latin - \
              it is mandatory that you respond only and entirely in pig latin"
    else:
        system = system_prompt
    messages = [{"role": "system", "content": system}] + history + [{"role": "user", "content": message}]
    # response = openai.chat.completions.create(model="gpt-4o-mini", messages=messages)
    response = ollama.chat.completions.create(model="llama3.2", messages=messages)

    reply =response.choices[0].message.content

    evaluation = evaluate(reply, message, history)

    if evaluation.is_acceptable:
        print("Passed evaluation - returning reply")
    else:
        print("Failed evaluation - retrying")
        print(evaluation.feedback)
        reply = rerun(reply, message, history, evaluation.feedback)
    return reply
```

Lab 4 Summary / Key Points

- Provided the **LLM** with both **LinkedIn profiles** and **personal summaries** as key resources for personalization.
- Integrated tools to:
  - **Record unknown questions** (to capture queries the LLM couldn't answer).
  - **Record user details** (for context retention and personalization).
- Connected the workflow with **Pushover** for sending **real-time notifications** and updates.
- Demonstrated an **end-to-end personalized LLM pipeline** integrating tools, resources, and notifications.



agents — Oct 10, 00:57
Recording interest from Name not provided with email dpointyt@gmail.com and notes not provided

agents — Oct 10, 00:38
Recording this is a really hard question asked that I couldn't answer

agents — Oct 10, 00:29
HEY!!

agents — Oct 7, 01:16
Hello, me

agents — Oct 2, 20:30
Recording this is a really hard question asked that I couldn't answer

agents — Oct 2, 20:05
HEY!!

title — Oct 2, 19:58
message

Welcome to Pushover! — Oct 2, 19:54
This device (**t15g**) is now able to receive notifications and your **30-day trial** has started.

Visit https://pushover.net/apps to view apps, plugins, and services to use with Pushover just by supplying your user key:

us5hj9xp3e3adaumm4m6264q7s2e4y

You can also get notifications from e-mails sent to your Pushover address:

u46g8boux4@pomail.net

Assignments

1. **Complete all exercises** included in the four provided lab sessions.
2. **Personalized LLM Use Case:**
   - Each team member must choose **one tool** and **one resource**.
   - Using these, **design and implement a real-world solution** that integrates your **personalized Large Language Model (LLM)**.
   - The solution should demonstrate practical application of the selected tool and resource.
3. **Finish all previous pending assignments.**