Workshop 1

Build a social media post generator

Andrew Mead, Vector Lab

January, 2026

Pre talk notes

- Mastadon account
    - Needs testing
        - Especially search and commenting on existing posts
- Openrouter account
- Specific needs ironing out

Placeholder for vibe coding session

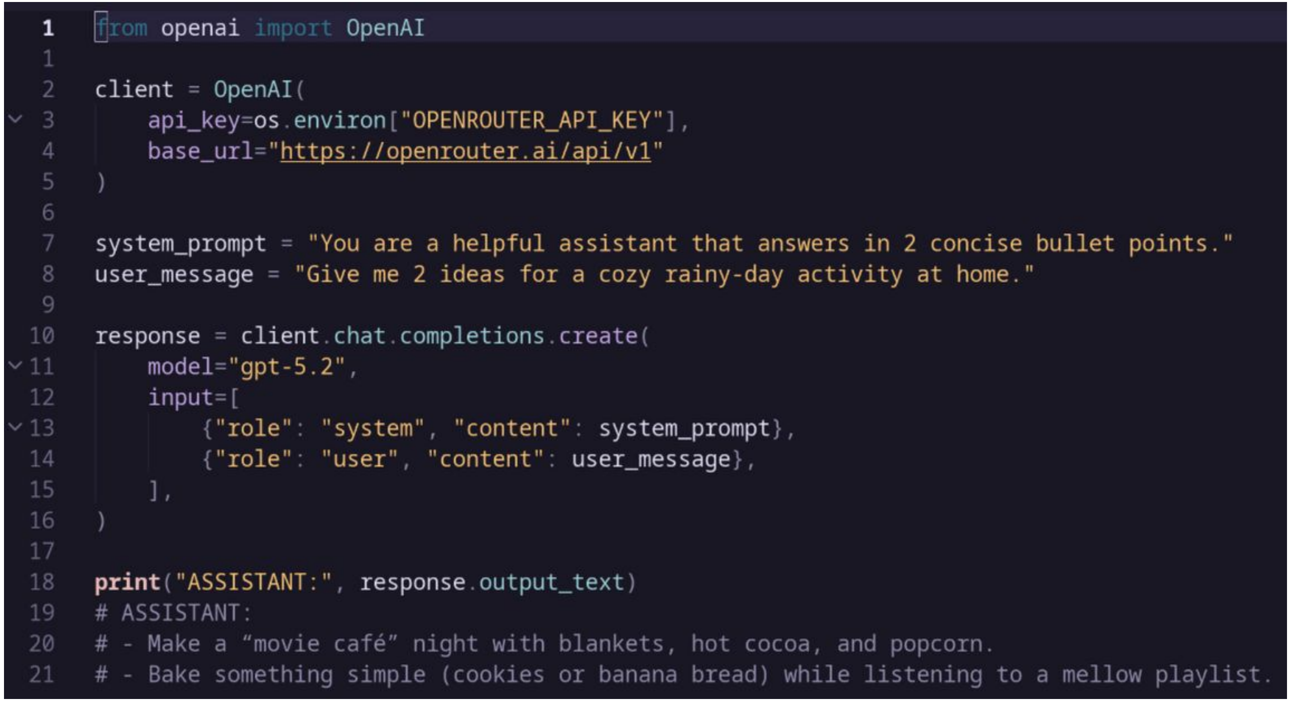
What we will be building today

- Generate real (or fake) docs about yourself or your company
- Use an LLM to generate social media posts
- Post on your social media account
- Get top article for your niche and generate comments for some of them

LLM Generation

- Use the OpenAI Api Library
  - Supported by pretty much all model providers
    - No lock in
    - You just need to change the base URL and API key
- Openrouter
  - Provider aggregator
    - One place to access all models
- Recommended to stay away from the OpenAI reponses API and direct Anthropic API library as it causes lock in

Basic LLM call with OpenRouter



```python
from openai import OpenAI

client = OpenAI(
    api_key=os.environ["OPENROUTER_API_KEY"],
    base_url="https://openrouter.ai/api/v1"
)

system_prompt = "You are a helpful assistant that answers in 2 concise bullet points."
user_message = "Give me 2 ideas for a cozy rainy-day activity at home."

response = client.chat.completions.create(
    model="gpt-5.2",
    input=[
        {"role": "system", "content": system_prompt},
        {"role": "user", "content": user_message},
    ],
)

print("ASSISTANT:", response.output_text)
# ASSISTANT:
# - Make a “movie café” night with blankets, hot cocoa, and popcorn.
# - Bake something simple (cookies or banana bread) while listening to a mellow playlist.
```

Structured outputs

- Often we want to have a deterministic structure to the LLM's responses
  - Called structured outputs
- Supported by the OpenAI API library
- JSON output

Structures outputs example

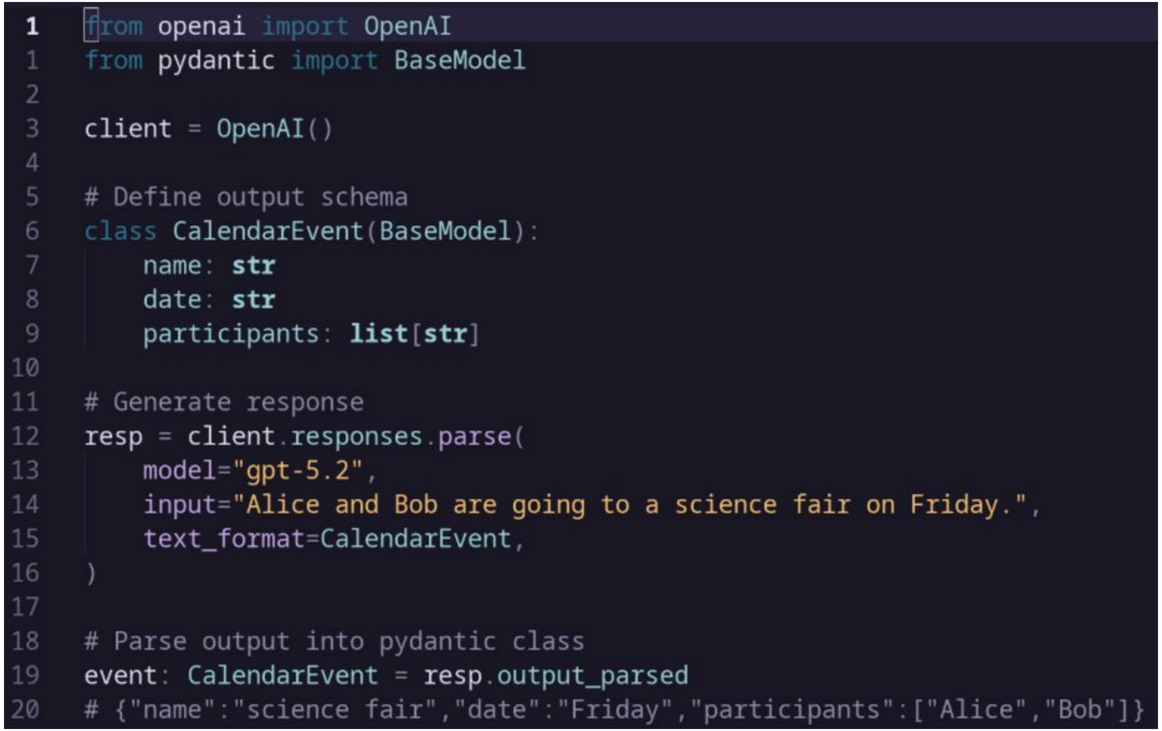
- We want to add events to a calendar based on natural language
- Example input: "Alice and Bob are going to a science fair on Friday."
- Desired output:

{"name": "science fair", "date": "Friday", "participants": ["Alice","Bob"]}

Pydantic

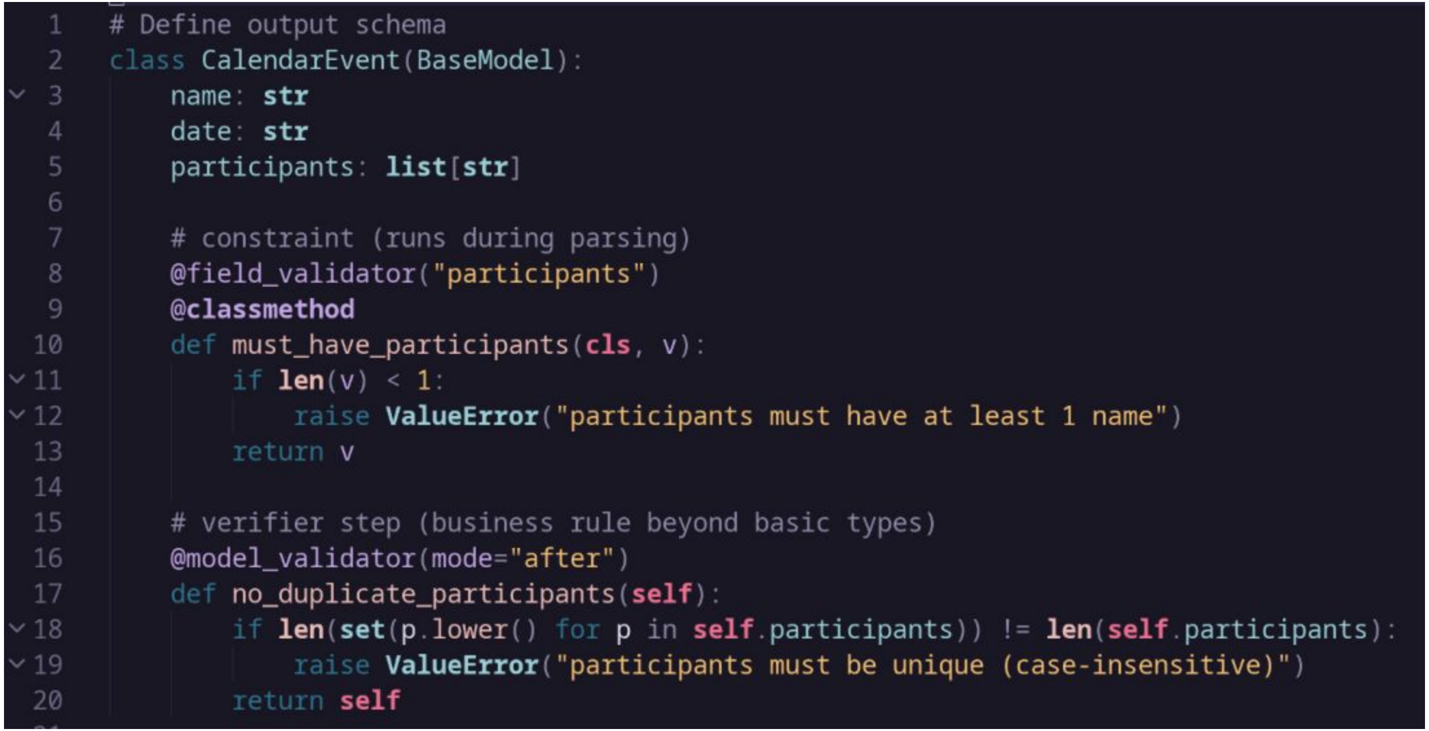
- Allows us to easily define structured outputs and parse them
- Directly integrates in with openAI Api library (supported by all models)
  - Use client.responses.parse() instead of client.chat.completions()
- Allows us to supply additional validators after the initial JSON is parse (i.e. making sure date format is correct, users exist in the db, etc)

Pydantic code example



```python
from openai import OpenAI
from pydantic import BaseModel

client = OpenAI()

# Define output schema
class CalendarEvent(BaseModel):
    name: str
    date: str
    participants: list[str]

# Generate response
resp = client.responses.parse(
    model="gpt-5.2",
    input="Alice and Bob are going to a science fair on Friday.",
    text_format=CalendarEvent,
)

# Parse output into pydantic class
event: CalendarEvent = resp.output_parsed
# {"name":"science fair","date":"Friday","participants":["Alice","Bob"]}
```

Pydantic Validators



```python
# Define output schema
class CalendarEvent(BaseModel):
    name: str
    date: str
    participants: list[str]

    # constraint (runs during parsing)
    @field_validator("participants")
    @classmethod
    def must_have_participants(cls, v):
        if len(v) < 1:
            raise ValueError("participants must have at least 1 name")
        return v

    # verifier step (business rule beyond basic types)
    @model_validator(mode="after")
    def no_duplicate_participants(self):
        if len(set(p.lower() for p in self.participants)) != len(self.participants):
            raise ValueError("participants must be unique (case-insensitive)")
        return self
```

Social Media Connection

- Mastadon
  - Not the sexiest or highest traffic, but is the easiest
  - Start with this to verify your AI pipeline is working as you would like, then integrate other services as desired
- Twitter, linkedin, Instagram, Whatsapp, etc
  - These are all much harder, as they dont have free, easily accessible public API

Misc Consideration

- Keep secrets secret
  - Use a .env file that is in .gitignore
- Use uv for python dependency management

Tasks for today

Goal 1:

- Make 3-5 docs describing your company, what you do, and what your brand is
  - Use an LLM to help you with this!

Goal 2:

- Feed these docs into an LLM that generates social media posts for your company

Tasks for today (pt 2)

Goal 3:

- Integrate with Mastadon to post the LLM made social media post

Goal 4:

- Get the 5 most recent posts for a keyword related to your business and make a reply to them
    - Use structured outputs to get responses to all of them at once

Tasks for today (optional)

(Optional) Goal 5: Integrate with other social media platforms