```
import os

import mermaid as md
from dotenv import load_dotenv
from loguru import logger
from mermaid.graph import Graph
from swarm_models import OpenAIChat
from swarms import Agent, extract_code_from_markdown
from uuid import uuid4

load_dotenv()

# Example with Groq
groq_api_key = os.getenv("GROQ_API_KEY")
model = OpenAIChat(
    openai_api_base="https://api.groq.com/openai/v1",
    openai_api_key=groq_api_key,
    model_name="llama-3.1-70b-versatile",
    temperature=0.1,
    max_tokens=4000,
)

GROWTH_STRATEGY_SYS_PROMPT = """
You're a hypergrowth strategist, your job is to take a business strategy for a product or company and generate a mermaid graph that outlines potential growth strategies focusing on hyper-growth
```

opportunities using blitzscaling techniques.

- **Input**: A detailed business strategy for a product or company.
- **Objective**: Identify and outline potential growth strategies focusing on hyper-growth opportunities using blitzscaling techniques.
- **Output**: A mermaid graph in syntax form which can be rendered in real-time.

Steps

1. **Analyze**: Review the provided business strategy to understand the current position, strengths, and goals of the product or company.
2. **Identify Growth Opportunities**: Utilize blitzscaling principles to pinpoint areas where rapid expansion is feasible. Consider market size, distribution channels, and technological innovations.
3. **Draft Strategy Elements**: Break down the overarching growth strategy into actionable elements or nodes. These should include key tactics, potential risks, and strategic pivots.
4. **Create Mermaid Graph**: Map out the identified nodes and connections, ensuring a clear narrative of the growth trajectory. Incorporate decision points, dependencies, and outcomes.

Output Format

The output should be in mermaid syntax, precisely formatted to ensure it can be rendered with mermaid tools. Ensure correct use of indentation and syntax for nodes, connections, and annotations.

Example

Input: "[Company X's strategic goal is to capture a significant share of the online education market by leveraging its existing technology platform while focusing on user acquisition, content partnerships, and international expansion.]"

Output:

```
graph TD;
    A[Start] --> B[Leverage Tech Platform];
    B --> C[User Acquisition];
    C --> D[Content Partnerships];
    D --> E[International Expansion];
    E --> F[Capture Market Share];
    F --> G[Evaluate and Iterate];
```

(The example above should be adjusted based on the specific input company strategy, involving more nodes and potential paths.)

Notes

- Ensure the strategy aligns with blitzscaling concepts: speed over efficiency, accepting risks, and focusing on winner-takes-all markets.
- Consider potential roadblocks and prepare bifurcation points within the graph where strategic adjustments might be necessary.
- The graph should visually narrate the strategy's progression and decision-making stages.
- Maintain flexibility to accommodate additional user input and iterate upon the presented strategic

model.

- The interaction should support the continuous refinement of the strategy and real-time updates to the Mermaid diagram.

- Only output the Mermaid graph syntax, nothing else.

- Always start with the word "```mermaid" and end with "```"

- Only output the Mermaid graph syntax, nothing else.

- Make sure make the graph as big as possible to see all the details.

"""

Initialize the agent

growth_strategy_agent = Agent(

    agent_name="Growth-Strategy-Agent",

    system_prompt=GROWTH_STRATEGY_SYS_PROMPT,

    llm=model,

    max_loops=1,

    autosave=True,

    dashboard=False,

    verbose=True,

    dynamic_temperature_enabled=True,

    saved_state_path="growth_strategy_agent.json",

    user_name="swarms_corp",

    retry_attempts=1,

    context_length=200000,

    return_step_meta=False,

    output_type="string",

    streaming_on=False,

```
    max_tokens=4000,
)



def generate_growth_strategy(agent: Agent, task: str, prev_graph: str = None):
    """
    Run the Tree of Thoughts agent and build on previous graph if provided.

    Args:
        agent (Agent): The agent to run
        task (str): The task to process
        prev_graph (str): Optional previous graph to build upon

    Returns:
        md.Mermaid: The rendered Mermaid graph
    """
    logger.info(f"Running Tree of Thoughts agent with task: {task}")

    if prev_graph:
        # Append new graph elements to previous graph
        logger.debug("Building on previous graph")
        graph = agent.run(task + f"\nPrevious graph:\n{prev_graph}")
        logger.debug(f"Generated graph: {graph}")
        print(graph)
    else:
```

```
        logger.debug("Generating new graph")
        graph = agent.run(task)
        logger.debug(f"Generated graph: {graph}")

    logger.info("Rendering final Mermaid graph")
    graph_code = extract_code_from_markdown(graph)

    graph = Graph('Sequence-diagram', graph_code)
    render = md.Mermaid(graph, width=3800, height=3000) # Increase size to see all details
    render.to_png(f"growth_strategy_graph_{uuid4()}.png") # Save the graph as an image

    logger.info(f"Saved graph to growth_strategy_graph_{uuid4()}.png")
    return render


generate_growth_strategy(growth_strategy_agent, "How can we grow a spreadsheet swarm product for b2b applications, it's a spreadsheet of a swarm of agents that all run concurrently. How do we grow this product ")
```