```
import json
import os
import uuid
from typing import List

from loguru import logger
from pydantic import BaseModel, Field

from swarms import (
    Agent,
    SpreadSheetSwarm,
)
from swarm_models import OpenAIChat
from swarm_models.openai_function_caller import OpenAIFunctionCaller

agent_pool = []


class AgentInfo(BaseModel):
    agent_name: str = Field(
        ...,
        title="Financial Agent",
    )
    # agent_description: str = Field(
    #     ...,
    #     description = "A financial agent that provides information on financial topics.",
```

```
    # )
    system_prompt: str = Field(
        ...,
        description="A custom system prompt for the agent that is very direct and provides instructions and multi-examples on how to complete it's task!",
    )
    # Max loops to run the agent
    # max_loops: int = Field(
    #     1,
    #     description="The maximum number of loops the agent will run for",
    # )
    # task: str = Field(
    #     ...,
    #     description="The very specific task that the agent is supposed to complete. This is the task that the user wants the agent to complete. Make it very specific and clear.",
    # )


class SwarmLevel(BaseModel):
    swarm_name: str = Field(
        ...,
        description="The name of the swarm.",
    )
    rules: str = Field(
        ...,
        description="Define the rules of engagement for the agents in the swarm.",
```

```
    )
    plan: str = Field(
        ...,
        description="Create a plan for the swarm, define the goal and then create a list of agents that are needed to accomplish the goal.",
    )
    # goal: str = Field(
    #     ...,
    #     description="The goal of the swarm. Extract the goal from the user's request.",
    # )
    task: str = Field(
        ...,
        description="The task that the user wants the swarm to complete. This is the task that the user wants the agents to complete. Make it very specific and clear.",
    )
    agents: List[AgentInfo] = Field(
        ...,
        description="The list of agents participating in the swarm.",
    )
    # communication_flow: str = Field(
    #     ...,
    #     description="""
    #     Define the communication flow between the agents in the swarm.
    #     Only output the names of the agents you have created, only output the flow of the agents followed by the name.
    #     Use the arrow sign for sequential processing and or a comma for parallel processing.
```

```
    #     Ensure that the names of the agents you pass are the same as the ones you have created,
    #     if there are agents you have not created, do not include them in the flow. Only include maybe 3 agents in the flow at a time.

    #     Follow this example Example:
    #     Agent Name 1 -> Agent Name 2 -> Agent Name 3

    #     """,
    # )

BOSS_SYS_PROMPT = """

You are a Director Boss Agent. Your primary responsibility is to manage a swarm of worker agents to efficiently, effectively, and skillfully serve the user.
You must decide whether to create new agents with specific capabilities or to delegate tasks to existing agents, ensuring that all operations are performed with maximum efficiency.

### Instructions:
1. **Task Assignment**:
   - When a task is presented, first analyze the available worker agents.
    - If an appropriate agent exists, delegate the task to them with clear, direct, and actionable instructions.
   - If no suitable agent exists, dynamically create a new agent with a fitting system prompt to handle the task.
```

2. **Agent Creation**:

- Name the agent according to the task it is intended to perform (e.g., "Twitter Marketing Agent").

- Provide the new agent with a concise and clear system prompt that includes its specific role, objectives, and any tools it can utilize.

3. **Efficiency**:

- Always strive to minimize redundancy and maximize task completion speed.

- Avoid unnecessary agent creation if an existing agent can fulfill the task.

4. **Communication**:

- When delegating tasks, be explicit in your instructions. Avoid ambiguity to ensure that agents understand and execute their tasks effectively.

- Ensure that agents report back on the completion of their tasks or if they encounter issues.

5. **Reasoning and Decisions**:

- Provide brief reasoning when selecting or creating agents for tasks to maintain transparency.

- Avoid using an agent if it is not necessary, and provide a clear explanation if no agents are suitable for a task.

"""

```
# Model
model = OpenAIFunctionCaller(
    system_prompt=BOSS_SYS_PROMPT,
    api_key=os.getenv("OPENAI_API_KEY"),
    base_model=SwarmLevel,
    max_tokens=5000,
```

```
)


def create_agents_with_boss(
    task: str,
):
    # Run the model
    out = model.run(task)
    # "Create a swarm of agents to provide financial advice on investing in stocks and bonds. Create a minimum of 10 agents that are hyper-specialized in different areas of financial advice. Define the rules of engagement for the agents in the swarm."

    print(out)
    print(type(out))

    # Make the dict look pretty
    pretty_out = json.dumps(out, indent=4)

    # Save the output to a file
    with open(f"{uuid.uuid4().hex}_agent_creation_dict", "w") as f:
        f.write(pretty_out)

    return out


def create_agents(
```

```
    agent_name: str,
    system_prompt: str,
    # task: str,
):
    # Create an instance of the OpenAIChat class
    llm_worker = OpenAIChat(
        api_key=os.getenv("OPENAI_API_KEY"),
        model_name="gpt-4o-mini",
        temperature=0.1,
    )

    agent_created = Agent(
        agent_name=agent_name,
        system_prompt=system_prompt,
        llm=llm_worker,
        max_loops=1,
        autosave=True,
        dashboard=False,
        verbose=True,
        dynamic_temperature_enabled=True,
        saved_state_path=f"{uuid.uuid4().hex}.json",
        user_name="swarms_corp",
        retry_attempts=1,
        context_length=128000,
    )
```

```
    return agent_created



def parse_agents_from_dict(
    input: dict,
) -> Agent:
    """
    Parses a dictionary containing agent information and creates Agent objects.

    Args:
        input (dict): A dictionary containing agent information.

    Returns:
        Agent: The created Agent object.

    """
    agents = input.get("agents")

    for agent in agents:
        agent_name = agent.get("agent_name")
        system_prompt = agent.get("system_prompt")

        agent = create_agents(
            agent_name=agent_name,
            system_prompt=system_prompt,
        )
```

```
        # Add to agent pool
        agent_pool.append(agent)

        logger.info(f"Agent {agent_name} created successfully.")



def build_and_run_swarm(agents: list, task: str) -> str:
    swarm = SpreadSheetSwarm(
        agents=agents,
        max_loops=3,
        save_file_path=f"spread_sheet_swarm{uuid.uuid4().hex}.csv",
    )

    return swarm.run(task)



def log_swarm_data(
    input: dict,
) -> None:
    name = input.get("swarm_name")
    rules = input.get("rules")
    plan = input.get("plan")

    logger.info(
        f"Swarm Name: {name} \n Rules: {rules} \n Plan: {plan} \n "
```

```
    )
    return None


def design_and_run_swarm(task: str = None):
    # First run the boss agent to get the dict of agents
    logger.info("Creating agents with the boss agent.")
    agents_dict = create_agents_with_boss(task)
    task_for_agent = agents_dict.get("task")

    # Log the swarm data
    log_swarm_data(agents_dict)

    # Parse the agents from the dict
    logger.info("Parsing agents from the dict.")
    parse_agents_from_dict(agents_dict)

    # Run the agents created by the boss
    logger.info("Running the agents.")

    return build_and_run_swarm(agent_pool, task_for_agent)


out = design_and_run_swarm(
    """
     Create a swarm of agents to that are specialized in every social media platform to market the
```

swarms github framework which makes it easy for you to orchestrate and manage multiple agents in a swarm.

Create a minimum of 10 agents that are hyper-specialized in different areas of social media marketing.

We need to promote the new SpreadSheet Swarm feature that allows you to run multiple agents in parallel and manage them from a single dashboard.

Here is the link: https://docs.swarms.world/en/latest/swarms/structs/spreadsheet_swarm/

"""

)

print(out)

out = design_and_run_swarm(

"""

Create a swarm of 10 agents that are specialized for sending api requests to OpenAI's API across major programming languages like Python, JavaScript, Go, Rust, and more

The agents must output the code snippet for sending an API request to OpenAI's API in the specified programming language.

"""

)

print(out)

```
# out = design_and_run_swarm(
#     """
#     Create a swarm of 10 agents that are specialized for sending api requests to OpenAI's API across major programming languages like Python, JavaScript, Go, Rust, and more
#     The agents must output the code snippet for sending an API request to OpenAI's API in the specified programming language.

#     """
# )
# print(out)


# out = design_and_run_swarm(
#     """
#     Create a swarm of 10 agents that are specialized to create games in python using various new game design patterns that are completely novel and unseen before.
#     The agent should output the python code for the their game and only their game.

#     Give them very novel and weird game ideas to implement, they should have different names

#     """
# )
# print(out)


# out = design_and_run_swarm(
```

```
# """
# Create a swarm of go to market strategists that are specialized in launching new products in swarms of agents.

# The agents must output a detailed go to market strategy for launching a new product in the market. The
# agents should be specialized with different perpsectives in business, marketing, sales, and product development.

# The product is a swarm of agents that can be used to orchestrate and manage multiple agents in a swarm. It's currently a gitbug repo that can be found here: github.com/kyegomez

# """
# )
# print(out)


# out = design_and_run_swarm(
# """
# Create a swarm of go to market strategists that are specialized in launching new products in swarms of agents.

# The agents must output a detailed go to market strategy for launching a new product in the market. The
# agents should be specialized with different perpsectives in business, marketing, sales, and product development.
```

```
#     The product is a swarm of agents that can be used to orchestrate and manage multiple agents in a swarm. It's currently a gitbug repo that can be found here: github.com/kyegomez

#     """
# )
# print(out)


# out = design_and_run_swarm(
#     """
#     Create a swarm of agents for producing music together.

#     One agent will be a songwriter, another one will be a marketing agent,
#     another one will be a music producer, another one will be a sound engineer,
#     another one will be a singer, another one will be a music video director,
#     another one will be a music video editor, another one will be a music video producer,
#     another one will be a music video marketer, another one will be a music video distributor.

#     Create a swarm of agents that are specialized in promoting this new music.

#     """
# )
# print(out)
```

```
# out = design_and_run_swarm(
#     """
#      Create a swarm of agents that will generate the python code to generate the qr codes for the following links:
#     Telegram: https://t.me/+Sm4J-sSkw8c0ODA5
#      Discord: https://discord.gg/F8sSH4Gh
#     https://lu.ma/GPTuesdays?k=c
#     """
# )
# print(out)
```