```
"""

Problem: We're creating specialized agents for various social medias

List of agents:
- Facebook agent
- Twitter agent
- Instagram agent
- LinkedIn agent
- TikTok agent
- Reddit agent
- Pinterest agent
- Snapchat agent
- YouTube agent
- WhatsApp agent

"""

from swarms import Agent, OpenAIChat, MixtureOfAgents
import os
import requests


# Model
model = OpenAIChat(max_tokens=4000, temperature=0.8)
```

```
# Content Variables
facebook_content = "Here is the content for Facebook"
twitter_content = "Here is the content for Twitter"
instagram_content = "Here is the content for Instagram"
linkedin_content = "Here is the content for LinkedIn"
tiktok_content = "Here is the content for TikTok"
reddit_content = "Here is the content for Reddit"
pinterest_content = "Here is the content for Pinterest"
snapchat_content = "Here is the content for Snapchat"
youtube_content = "Here is the content for YouTube"
whatsapp_content = "Here is the content for WhatsApp"

# Prompt Variables
facebook_prompt = f"""
You are a Facebook social media agent. Your task is to create a post that maximizes engagement
on Facebook. Use rich media, personal stories, and interactive content. Ensure the post is
compelling and includes a call-to-action. Here is the content to work with: {facebook_content}
"""

twitter_prompt = f"""
You are a Twitter social media agent. Your task is to create a tweet that is short, concise, and uses
trending hashtags. The tweet should be engaging and include relevant media such as images, GIFs,
or short videos. Here is the content to work with: {twitter_content}
"""

instagram_prompt = f"""
```

```
You are an Instagram social media agent. Your task is to create a visually appealing post that includes high-quality images and engaging captions. Consider using stories and reels to maximize reach. Here is the content to work with: {instagram_content}
"""

linkedin_prompt = f"""
You are a LinkedIn social media agent. Your task is to create a professional and insightful post related to industry trends or personal achievements. The post should include relevant media such as articles, professional photos, or videos. Here is the content to work with: {linkedin_content}
"""

tiktok_prompt = f"""
You are a TikTok social media agent. Your task is to create a short, entertaining video that aligns with trending challenges and music. The video should be engaging and encourage viewers to interact. Here is the content to work with: {tiktok_content}
"""

reddit_prompt = f"""
You are a Reddit social media agent. Your task is to create an engaging post for relevant subreddits. The post should spark in-depth discussions and include relevant media such as images or links. Here is the content to work with: {reddit_content}
"""

pinterest_prompt = f"""
You are a Pinterest social media agent. Your task is to create high-quality, visually appealing pins. Focus on popular categories such as DIY, fashion, and lifestyle. Here is the content to work with:
```

```
{pinterest_content}
"""


snapchat_prompt = f"""
You are a Snapchat social media agent. Your task is to create engaging and timely snaps and
stories. Include personal touches and use filters or AR lenses to enhance the content. Here is the
content to work with: {snapchat_content}
"""


youtube_prompt = f"""
You are a YouTube social media agent. Your task is to create high-quality videos with engaging
thumbnails. Ensure a consistent posting schedule and encourage viewer interaction. Here is the
content to work with: {youtube_content}
"""


whatsapp_prompt = f"""
You are a WhatsApp social media agent. Your task is to send personalized messages and updates.
Use broadcast lists and ensure the messages are engaging and relevant. Here is the content to
work with: {whatsapp_content}
"""



def post_to_twitter(content: str) -> None:
    """
    Posts content to Twitter.
```

```
Args:
    content (str): The content to post on Twitter.

Raises:
    ValueError: If the content is empty or exceeds the character limit.
    requests.exceptions.RequestException: If there is an error with the request.
"""
try:
    if not content:
        raise ValueError("Content cannot be empty.")
    if len(content) > 280:
        raise ValueError(
            "Content exceeds Twitter's 280 character limit."
        )

    # Retrieve the access token from environment variables
    access_token = os.getenv("TWITTER_ACCESS_TOKEN")
    if not access_token:
        raise EnvironmentError(
            "Twitter access token not found in environment variables."
        )

    # Mock API endpoint for example purposes
    api_url = "https://api.twitter.com/2/tweets"
    headers = {
        "Authorization": f"Bearer {access_token}",
            "Content-Type": "application/json",
        }
        data = {"text": content}
        response = requests.post(api_url, headers=headers, json=data)
        response.raise_for_status()

        print("Content posted to Twitter successfully.")
    except ValueError as e:
        print(f"Error: {e}")
        raise
    except requests.exceptions.RequestException as e:
        print(f"Error: {e}")
        raise


def post_to_instagram(content: str) -> None:
    """
    Posts content to Instagram.

    Args:
        content (str): The content to post on Instagram.

    Raises:
        ValueError: If the content is empty or exceeds the character limit.
        requests.exceptions.RequestException: If there is an error with the request.
    """
    try:
        if not content:
            raise ValueError("Content cannot be empty.")
        if len(content) > 2200:
            raise ValueError(
                "Content exceeds Instagram's 2200 character limit."
            )

        # Retrieve the access token from environment variables
        access_token = os.getenv("INSTAGRAM_ACCESS_TOKEN")
        user_id = os.getenv("INSTAGRAM_USER_ID")
        if not access_token or not user_id:
            raise EnvironmentError(
                "Instagram access token or user ID not found in environment variables."
            )

        # Mock API endpoint for example purposes
        api_url = f"https://graph.instagram.com/v10.0/{user_id}/media"
        headers = {
            "Authorization": f"Bearer {access_token}",
            "Content-Type": "application/json",
        }
        data = {
            "caption": content,
                "image_url": "URL_OF_THE_IMAGE_TO_POST",  # Replace with actual image URL if
needed
```

```
        }
        response = requests.post(api_url, headers=headers, json=data)
        response.raise_for_status()

        print("Content posted to Instagram successfully.")
    except ValueError as e:
        print(f"Error: {e}")
        raise
    except requests.exceptions.RequestException as e:
        print(f"Error: {e}")
        raise


def post_to_facebook(content: str) -> None:
    """
    Posts content to Facebook.

    Args:
        content (str): The content to post on Facebook.

    Raises:
        ValueError: If the content is empty.
        requests.exceptions.RequestException: If there is an error with the request.
    """
    try:
        if not content:
```

```
        raise ValueError("Content cannot be empty.")

    # Retrieve the access token from environment variables
    access_token = os.getenv("FACEBOOK_ACCESS_TOKEN")
    if not access_token:
        raise EnvironmentError(
            "Facebook access token not found in environment variables."
        )

    # Mock API endpoint for example purposes
    api_url = "https://graph.facebook.com/v10.0/me/feed"
    headers = {
        "Authorization": f"Bearer {access_token}",
        "Content-Type": "application/json",
    }
    data = {"message": content}
    response = requests.post(api_url, headers=headers, json=data)
    response.raise_for_status()

    print("Content posted to Facebook successfully.")
except ValueError as e:
    print(f"Error: {e}")
    raise
except requests.exceptions.RequestException as e:
    print(f"Error: {e}")
    raise
```

```
# Prompts
prompts = [
    facebook_prompt,
    twitter_prompt,
    instagram_prompt,
    linkedin_prompt,
    tiktok_prompt,
    reddit_prompt,
    pinterest_prompt,
    snapchat_prompt,
    youtube_prompt,
    whatsapp_prompt,
]


# For every prompt, we're going to create a list of agents
for prompt in prompts:
    agents = [
        Agent(
            agent_name="Facebook Agent",
            system_prompt=prompt,
            llm=model,
            max_loops=1,
            dashboard=False,
    streaming_on=True,
    verbose=True,
    dynamic_temperature_enabled=True,
    stopping_token="<DONE>",
    state_save_file_type="json",
    saved_state_path="facebook_agent.json",
),
Agent(
    agent_name="Twitter Agent",
    system_prompt=prompt,
    llm=model,
    max_loops=1,
    dashboard=False,
    streaming_on=True,
    verbose=True,
    tools=[post_to_twitter],
    dynamic_temperature_enabled=True,
    stopping_token="<DONE>",
    state_save_file_type="json",
    saved_state_path="twitter_agent.json",
),
Agent(
    agent_name="Instagram Agent",
    system_prompt=prompt,
    llm=model,
    max_loops=1,
    dashboard=False,
    streaming_on=True,
    verbose=True,
    dynamic_temperature_enabled=True,
    stopping_token="<DONE>",
    tools=[post_to_instagram],
    state_save_file_type="json",
    saved_state_path="instagram_agent.json",
),
Agent(
    agent_name="LinkedIn Agent",
    system_prompt=prompt,
    llm=model,
    max_loops=1,
    dashboard=False,
    streaming_on=True,
    verbose=True,
    dynamic_temperature_enabled=True,
    stopping_token="<DONE>",
    state_save_file_type="json",
    saved_state_path="linkedin_agent.json",
),
Agent(
    agent_name="TikTok Agent",
    system_prompt=prompt,
    llm=model,
    max_loops=1,
    dashboard=False,
    streaming_on=True,
    verbose=True,
    dynamic_temperature_enabled=True,
    stopping_token="<DONE>",
    state_save_file_type="json",
    saved_state_path="tiktok_agent.json",
),
Agent(
    agent_name="Reddit Agent",
    system_prompt=prompt,
    llm=model,
    max_loops=1,
    dashboard=False,
    streaming_on=True,
    verbose=True,
    dynamic_temperature_enabled=True,
    stopping_token="<DONE>",
    state_save_file_type="json",
    saved_state_path="reddit_agent.json",
),
Agent(
    agent_name="Pinterest Agent",
    system_prompt=prompt,
    llm=model,
            max_loops=1,
            dashboard=False,
            streaming_on=True,
            verbose=True,
            dynamic_temperature_enabled=True,
            stopping_token="<DONE>",
            state_save_file_type="json",
            saved_state_path="pinterest_agent.json",
        ),
        Agent(
            agent_name="Snapchat Agent",
            system_prompt=prompt,
            llm=model,
            max_loops=1,
            dashboard=False,
            streaming_on=True,
            verbose=True,
            dynamic_temperature_enabled=True,
            stopping_token="<DONE>",
            state_save_file_type="json",
            saved_state_path="snapchat_agent.json",
        ),
    ]


# Final agent
```

```
final_agent = Agent(
    agent_name="Final Agent",
    system_prompt="Ensure the content is optimized for all social media platforms.",
    llm=model,
    max_loops=1,
    dashboard=False,
    streaming_on=True,
    verbose=True,
    dynamic_temperature_enabled=True,
    stopping_token="<DONE>",
    state_save_file_type="json",
    saved_state_path="final_agent.json",
)



# Create a mixture of agents
swarm = MixtureOfAgents(
    agents=agents,
    final_agent=final_agent,
    layers=1,
    verbose=True,
)

# parallel_swarm = AgentRearrange(
#     agents=agents,
#     flow=f"{agents[0].agent_name} -> {agents[1].agent_name}, {agents[2].agent_name},
{agents[3].agent_name},  {agents[4].agent_name}, {agents[5].agent_name}",
#     max_loops=1,
#     verbose=True,
# )


# Run the swarm
swarm.run(
    """



[Workshop Today][Unlocking The Secrets of Multi-Agent Collaboration]


[Location][https://lu.ma/tfn0fp37]
[Time][Today 2:30pm PST -> 4PM PST] [Circa 5 hours]


Sign up and invite your friends we're going to dive into various multi-agent orchestration workflows
in swarms:
https://github.com/kyegomez/swarms


And, the swarms docs:
https://docs.swarms.world/en/latest/



"""
)
```