SpreadSheetSwarm Documentation

Class Definition

```python
class SpreadSheetSwarm:
```

Full Path

```python
from swarms.structs.spreadsheet_swarm import SpreadSheetSwarm
```

Attributes

The `SpreadSheetSwarm` class contains several attributes that define its behavior and configuration. These attributes are initialized in the constructor (`__init__` method) and are used throughout the class to manage the swarm's operations.

Attribute	Type	Description

`name`	`str`	The name of the swarm.
`description`	`str`	A description of the swarm's purpose.
`agents`	`Union[Agent, List[Agent]]`	The agents participating in the swarm. Can be a single agent or a list of agents.
`autosave_on`	`bool`	Flag indicating whether autosave is enabled.
`save_file_path`	`str`	The file path where the swarm data will be saved.
`task_queue`	`queue.Queue`	The queue that stores tasks to be processed by the agents.
`lock`	`threading.Lock`	A lock used for thread synchronization to prevent race conditions.
`metadata`	`SwarmRunMetadata`	Metadata for the swarm run, including start time, end time, tasks completed, and outputs.
`run_all_agents`	`bool`	Flag indicating whether to run all agents or just one.
`max_loops`	`int`	The number of times to repeat the task.
`workspace_dir`	`str`	The directory where the workspace is located, retrieved from environment variables.

Parameters

- **`name`** (`str`, optional): The name of the swarm. Default is `"Spreadsheet-Swarm"`.

- **`description`** (`str`, optional): A brief description of the swarm. Default is `"A swarm that processes tasks from a queue using multiple agents on different threads."`.
- **`agents`** (`Union[Agent, List[Agent]]`, optional): The agents participating in the swarm. Default is an empty list.
- **`autosave_on`** (`bool`, optional): A flag to indicate if autosave is enabled. Default is `True`.
- **`save_file_path`** (`str`, optional): The file path where swarm data will be saved. Default is `"spreedsheet_swarm.csv"`.
- **`run_all_agents`** (`bool`, optional): Flag to determine if all agents should run. Default is `True`.
- **`max_loops`** (`int`, optional): The number of times to repeat the task. Default is `1`.
- **`workspace_dir`** (`str`, optional): The directory where the workspace is located. Default is retrieved from environment variable `WORKSPACE_DIR`.

Constructor (`__init__`)

The constructor initializes the `SpreadSheetSwarm` with the provided parameters. It sets up the task queue, locks for thread synchronization, and initializes the metadata.

Methods

`reliability_check`

```python
def reliability_check(self):
```

Description

The `reliability_check` method performs a series of checks to ensure that the swarm is properly configured before it begins processing tasks. It verifies that there are agents available and that a valid file path is provided for saving the swarm's data. If any of these checks fail, an exception is raised.

Raises

- **`ValueError`**: Raised if no agents are provided or if no save file path is specified.

Example

```python
swarm = SpreadSheetSwarm(agents=[agent1, agent2])
swarm.reliability_check()
```

`run`

```python
def run(self, task: str, *args, **kwargs):
```

Description

The `run` method starts the task processing using the swarm. Depending on the configuration, it can either run all agents or a specific subset of them. The method tracks the start and end times of the task, executes the task multiple times if specified, and logs the results.

Parameters

- **`task`** (`str`): The task to be executed by the swarm.
- **`*args`**: Additional positional arguments to pass to the agents.
- **`**kwargs`**: Additional keyword arguments to pass to the agents.

Example

```python
swarm = SpreadSheetSwarm(agents=[agent1, agent2])
swarm.run("Process Data")
```

`export_to_json`

```python
def export_to_json(self):
```

```

#### Description

The `export_to_json` method generates a JSON representation of the swarm's metadata. This can be useful for exporting the results to an external system or for logging purposes.

#### Returns

- **`str`**: The JSON representation of the swarm's metadata.

#### Example

```python
json_data = swarm.export_to_json()
print(json_data)
```

`data_to_json_file`

```python
def data_to_json_file(self):
```
```

Description

The `data_to_json_file` method saves the swarm's metadata as a JSON file in the specified workspace directory. The file name is generated using the swarm's name and run ID.

Example

```python
swarm.data_to_json_file()
```

`_track_output`

```python
def _track_output(self, agent: Agent, task: str, result: str):
```

Description

The `_track_output` method is used internally to record the results of tasks executed by the agents. It updates the metadata with the completed tasks and their results.

Parameters

- **`agent`** (`Agent`): The agent that executed the task.

- **`task`** (`str`): The task that was executed.

- **`result`** (`str`): The result of the task execution.

Example

```python
swarm._track_output(agent1, "Process Data", "Success")
```

`_save_to_csv`

```python
def _save_to_csv(self):
```

Description

The `_save_to_csv` method saves the swarm's metadata to a CSV file. It logs each task and its result before writing them to the file. The file is saved in the location specified by `save_file_path`.

Example

```python
```

swarm._save_to_csv()

```

---

## Usage Examples

### Example 1: Basic Swarm Initialization

```python
import os

from swarms import Agent
from swarm_models import OpenAIChat
from swarms.prompts.finance_agent_sys_prompt import (
    FINANCIAL_AGENT_SYS_PROMPT,
)
from swarms.structs.spreadsheet_swarm import SpreadSheetSwarm

# Example usage:
api_key = os.getenv("OPENAI_API_KEY")

# Model
model = OpenAIChat(
    openai_api_key=api_key, model_name="gpt-4o-mini", temperature=0.1
)
```
```

```
# Initialize your agents (assuming the Agent class and model are already defined)
agents = [
    Agent(
        agent_name=f"Financial-Analysis-Agent-spreesheet-swarm:{i}",
        system_prompt=FINANCIAL_AGENT_SYS_PROMPT,
        llm=model,
        max_loops=1,
        dynamic_temperature_enabled=True,
        saved_state_path="finance_agent.json",
        user_name="swarms_corp",
        retry_attempts=1,
    )
    for i in range(10)
]

# Create a Swarm with the list of agents
swarm = SpreadSheetSwarm(
    name="Finance-Spreadsheet-Swarm",
      description="A swarm that processes tasks from a queue using multiple agents on different threads.",
    agents=agents,
    autosave_on=True,
    save_file_path="financial_spreed_sheet_swarm_demo.csv",
    run_all_agents=False,
```

```
    max_loops=1,
)

# Run the swarm
swarm.run(
    task="Analyze the states with the least taxes for LLCs. Provide an overview of all tax rates and add them with a comprehensive analysis"
)
```

Example 2: QR Code Generator

```python
import os
from swarms import Agent
from swarm_models import OpenAIChat
from swarms.structs.spreadsheet_swarm import SpreadSheetSwarm

# Define custom system prompts for QR code generation
QR_CODE_AGENT_1_SYS_PROMPT = """
You are a Python coding expert. Your task is to write a Python script to generate a QR code for the link: https://lu.ma/jjc1b2bo. The code should save the QR code as an image file.
"""

QR_CODE_AGENT_2_SYS_PROMPT = """
```

You are a Python coding expert. Your task is to write a Python script to generate a QR code for the link: https://github.com/The-Swarm-Corporation/Cookbook. The code should save the QR code as an image file.

"""

```
# Example usage:
api_key = os.getenv("OPENAI_API_KEY")

# Model
model = OpenAIChat(
    openai_api_key=api_key, model_name="gpt-4o-mini", temperature=0.1
)

# Initialize your agents for QR code generation
agents = [
    Agent(
        agent_name="QR-Code-Generator-Agent-Luma",
        system_prompt=QR_CODE_AGENT_1_SYS_PROMPT,
        llm=model,
        max_loops=1,
        dynamic_temperature_enabled=True,
        saved_state_path="qr_code_agent_luma.json",
        user_name="swarms_corp",
        retry_attempts=1,
    ),
    Agent(
```

```
        agent_name="QR-Code-Generator-Agent-Cookbook",
        system_prompt=QR_CODE_AGENT_2_SYS_PROMPT,
        llm=model,
        max_loops=1,
        dynamic_temperature_enabled=True,
        saved_state_path="qr_code_agent_cookbook.json",
        user_name="swarms_corp",
        retry_attempts=1,
    ),
]

# Create a Swarm with the list of agents
swarm = SpreadSheetSwarm(
    name="QR-Code-Generation-Swarm",
    description="A swarm that generates Python scripts to create QR codes for specific links.",
    agents=agents,
    autosave_on=True,
    save_file_path="qr_code_generation_results.csv",
    run_all_agents=False,
    max_loops=1,
)

# Run the swarm
swarm.run(
    task="Generate Python scripts to create QR codes for the provided links and save them as image files."
```

)

```

## Example 3: Social Media Marketing

```python

import os

from swarms import Agent

from swarm_models import OpenAIChat

from swarms.structs.spreadsheet_swarm import SpreadSheetSwarm

# Define custom system prompts for each social media platform

TWITTER_AGENT_SYS_PROMPT = """

You are a Twitter marketing expert. Your task is to create engaging, concise tweets and analyze trends to maximize engagement. Consider hashtags, timing, and content relevance.

"""

INSTAGRAM_AGENT_SYS_PROMPT = """

You are an Instagram marketing expert. Your task is to create visually appealing and engaging content, including captions and hashtags, tailored to a specific audience.

"""

FACEBOOK_AGENT_SYS_PROMPT = """

You are a Facebook marketing expert. Your task is to craft posts that are optimized for engagement

```
and reach on Facebook, including using images, links, and targeted messaging.
"""

EMAIL_AGENT_SYS_PROMPT = """
You are an Email marketing expert. Your task is to write compelling email campaigns that drive conversions, focusing on subject lines, personalization, and call-to-action strategies.
"""

# Example usage:
api_key = os.getenv("OPENAI_API_KEY")

# Model
model = OpenAIChat(
    openai_api_key=api_key, model_name="gpt-4o-mini", temperature=0.1
)

# Initialize your agents for different social media platforms
agents = [
    Agent(
        agent_name="Twitter-Marketing-Agent",
        system_prompt=TWITTER_AGENT_SYS_PROMPT,
        llm=model,
        max_loops=1,
        dynamic_temperature_enabled=True,
        saved_state_path="twitter_agent.json",
        user_name="swarms_corp",
```

```
    retry_attempts=1,
),
Agent(
    agent_name="Instagram-Marketing-Agent",
    system_prompt=INSTAGRAM_AGENT_SYS_PROMPT,
    llm=model,
    max_loops=1,
    dynamic_temperature_enabled=True,
    saved_state_path="instagram_agent.json",
    user_name="swarms_corp",
    retry_attempts=1,
),
Agent(
    agent_name="Facebook-Marketing-Agent",
    system_prompt=FACEBOOK_AGENT_SYS_PROMPT,
    llm=model,
    max_loops=1,
    dynamic_temperature_enabled=True,
    saved_state_path="facebook_agent.json",
    user_name="swarms_corp",
    retry_attempts=1,
),
Agent(
    agent_name="Email-Marketing-Agent",
    system_prompt=EMAIL_AGENT_SYS_PROMPT,
    llm=model,
```

```
        max_loops=1,
        dynamic_temperature_enabled=True,
        saved_state_path="email_agent.json",
        user_name="swarms_corp",
        retry_attempts=1,
    ),
]

# Create a Swarm with the list of agents
swarm = SpreadSheetSwarm(
    name="Social-Media-Marketing-Swarm",
      description="A swarm that processes social media marketing tasks using multiple agents on different threads.",
    agents=agents,
    autosave_on=True,
    save_file_path="social_media_marketing_spreadsheet.csv",
    run_all_agents=False,
    max_loops=2,
)

# Run the swarm
swarm.run(
     task="Create posts to promote hack nights in miami beach for developers, engineers, and tech enthusiasts. Include relevant hashtags, images, and engaging captions."
)
```

Additional Information and Tips

- **Thread Synchronization**: When working with multiple agents in a concurrent environment, it's crucial to ensure that access to shared resources is properly synchronized using locks to avoid race conditions.

- **Autosave Feature**: If you enable the `autosave_on` flag, ensure that the file path provided is correct and writable. This feature is handy for long-running tasks where you want to periodically save the state.

- **Error Handling**

: Implementing proper error handling within your agents can prevent the swarm from crashing during execution. Consider catching exceptions in the `run` method and logging errors appropriately.

- **Custom Agents**: You can extend the `Agent` class to create custom agents that perform specific tasks tailored to your application's needs.

References and Resources

- [Python's `queue` module](https://docs.python.org/3/library/queue.html)

- [Python's `threading` module](https://docs.python.org/3/library/threading.html)

- [CSV File Handling in Python](https://docs.python.org/3/library/csv.html)

- [JSON Handling in Python](https://docs.python.org/3/library/json.html)