```
from pydantic.v1 import BaseModel
from typing import List, Callable
from swarms.utils.loguru_logger import initialize_logger

logger = initialize_logger(log_folder="swarm_registry")


class SwarmRegistry(BaseModel):
    swarm_pool: List[Callable] = []

    def add(self, swarm: Callable, *args, **kwargs):
        """
        Adds a swarm to the registry.

        Args:
            swarm (Callable): The swarm to add to the registry.
        """
        self.swarm_pool.append(swarm, *args, **kwargs)

    def query(self, swarm_name: str) -> Callable:
        """
        Queries the registry for a swarm by name.

        Args:
            swarm_name (str): The name of the swarm to query.
```

```
        Returns:
            Callable: The swarm function corresponding to the given name.
        """
        if not self.swarm_pool:
            raise ValueError("No swarms found in registry")

        if not swarm_name:
            raise ValueError("No swarm name provided.")

        for swarm in self.swarm_pool:
            if swarm.__name__ == swarm_name:
                name = swarm.__name__
                description = (
                    swarm.__doc__.strip().split("\n")[0]
                    or swarm.description
                )
                agent_count = len(swarm.agents)
                task_count = len(swarm.tasks)

                log = f"Swarm: {name}\nDescription: {description}\nAgents: {agent_count}\nTasks: {task_count}"
                logger.info(log)

                return swarm

        raise ValueError(
```

```
        f"Swarm '{swarm_name}' not found in registry."
    )

def remove(self, swarm_name: str):
    """
    Removes a swarm from the registry by name.

    Args:
        swarm_name (str): The name of the swarm to remove.
    """
    for swarm in self.swarm_pool:
        if swarm.__name__ == swarm_name:
            self.swarm_pool.remove(swarm)
            return
    raise ValueError(
        f"Swarm '{swarm_name}' not found in registry."
    )

def list_swarms(self) -> List[str]:
    """
    Lists the names of all swarms in the registry.

    Returns:
        List[str]: A list of swarm names.
    """
    if not self.swarm_pool:
```

```
            raise ValueError("No swarms found in registry.")

        for swarm in self.swarm_pool:
            name = swarm.__name__
            description = (
                swarm.__doc__.strip().split("\n")[0]
                or swarm.description
            )
            agent_count = len(swarm.agents)
            task_count = len(swarm.tasks)

                    log = f"Swarm: {name}\nDescription: {description}\nAgents: {agent_count}\nTasks:
{task_count}"
            logger.info(log)

        return [swarm.__name__ for swarm in self.swarm_pool]

    def run(self, swarm_name: str, *args, **kwargs):
        """
        Runs a swarm by name with the given arguments.

        Args:
            swarm_name (str): The name of the swarm to run.
            *args: Variable length argument list.
            **kwargs: Arbitrary keyword arguments.
```

```
    Returns:
        Any: The result of running the swarm.
    """
    swarm = self.query(swarm_name)
    return swarm(*args, **kwargs)

def add_list_of_swarms(self, swarms: List[Callable]):
    """
    Adds a list of swarms to the registry.

    Args:
        swarms (List[Callable]): A list of swarms to add to the registry.
    """
    for swarm in swarms:
        self.add(swarm)

    return self.swarm_pool

def query_multiple_of_swarms(
    self, swarm_names: List[str]
) -> List[Callable]:
    """
    Queries the registry for multiple swarms by name.

    Args:
        swarm_names (List[str]): A list of swarm names to query.
```

```
    Returns:
        List[Callable]: A list of swarm functions corresponding to the given names.
    """
    return [self.query(swarm_name) for swarm_name in swarm_names]

def remove_list_of_swarms(self, swarm_names: List[str]):
    """
    Removes a list of swarms from the registry by name.

    Args:
        swarm_names (List[str]): A list of swarm names to remove.
    """
    for swarm_name in swarm_names:
        self.remove(swarm_name)

    return self.swarm_pool

def run_multiple_of_swarms(
    self, swarm_names: List[str], *args, **kwargs
):
    """
    Runs a list of swarms by name with the given arguments.

    Args:
        swarm_names (List[str]): A list of swarm names to run.
```

```
            *args: Variable length argument list.
            **kwargs: Arbitrary keyword arguments.

        Returns:
            List[Any]: A list of results of running the swarms.
        """
        return [
            self.run(swarm_name, *args, **kwargs)
            for swarm_name in swarm_names
        ]


# Decorator to add a function to the registry
def swarm_registry():
    """
    Decorator to add a function to the registry.

    Args:
        swarm_registry (SwarmRegistry): The swarm registry instance.

    Returns:
        Callable: The decorated function.
    """

    def decorator(func, *args, **kwargs):
        try:
```

```
        swarm_registry = SwarmRegistry()
        swarm_registry.add(func, *args, **kwargs)
        logger.info(
            f"Added swarm '{func.__name__}' to the registry."
        )
        return func
    except Exception as e:
        logger.error(str(e))
        raise


return decorator
```