```
import requests
import datetime
from typing import List, Dict, Tuple
from loguru import logger
from swarms import Agent
from swarm_models import OpenAIChat

# GitHub API Configurations
GITHUB_REPO = "kyegomez/swarms"  # Swarms GitHub repository
GITHUB_API_URL = f"https://api.github.com/repos/{GITHUB_REPO}/commits"

# Initialize Loguru
logger.add(
    "commit_summary.log",
    rotation="1 MB",
    level="INFO",
    backtrace=True,
    diagnose=True,
)


# Step 1: Fetch the latest commits from GitHub
def fetch_latest_commits(
    repo_url: str, limit: int = 5
) -> List[Dict[str, str]]:
    """
```

```
Fetch the latest commits from a public GitHub repository.
"""
logger.info(
    f"Fetching the latest {limit} commits from {repo_url}"
)
try:
    params = {"per_page": limit}
    response = requests.get(repo_url, params=params)
    response.raise_for_status()

    commits = response.json()
    commit_data = []

    for commit in commits:
        commit_data.append(
            {
                "sha": commit["sha"][:7],  # Short commit hash
                "author": commit["commit"]["author"]["name"],
                "message": commit["commit"]["message"],
                "date": commit["commit"]["author"]["date"],
            }
        )

    logger.success("Successfully fetched commit data")
    return commit_data
```

```
    except Exception as e:
        logger.error(f"Error fetching commits: {e}")
        raise



# Step 2: Format commits and fetch current time
def format_commits_with_time(
    commits: List[Dict[str, str]]
) -> Tuple[str, str]:
    """
    Format commit data into a readable string and return current time.
    """
    current_time = datetime.datetime.now().strftime(
        "%Y-%m-%d %H:%M:%S"
    )
    logger.info(f"Formatting commits at {current_time}")

    commit_summary = "\n".join(
        [
            f"- `{commit['sha']}` by {commit['author']} on {commit['date']}: {commit['message']}"
            for commit in commits
        ]
    )

    logger.success("Commits formatted successfully")
    return current_time, commit_summary
```

```
# Step 3: Build a dynamic system prompt
def build_custom_system_prompt(
    current_time: str, commit_summary: str
) -> str:
    """
    Build a dynamic system prompt with the current time and commit summary.
    """
    logger.info("Building the custom system prompt for the agent")
    prompt = f"""
You are a software analyst tasked with summarizing the latest commits from the Swarms GitHub repository.

The current time is **{current_time}**.

Here are the latest commits:
{commit_summary}

**Your task**:
1. Summarize the changes into a clear and concise table in **markdown format**.
2. Highlight the key improvements and fixes.
3. End your output with the token `<DONE>`.

Make sure the table includes the following columns: Commit SHA, Author, Date, and Commit Message.
```

```
    """
    logger.success("System prompt created successfully")
    return prompt


# Step 4: Initialize the Agent
def initialize_agent() -> Agent:
    """
    Initialize the Swarms agent with OpenAI model.
    """
    logger.info("Initializing the agent with GPT-4o")
    model = OpenAIChat(model_name="gpt-4o")

    agent = Agent(
        agent_name="Commit-Summarization-Agent",
        agent_description="Fetch and summarize GitHub commits for Swarms repository.",
        system_prompt="",  # Will set dynamically
        max_loops=1,
        llm=model,
        dynamic_temperature_enabled=True,
        user_name="Kye",
        retry_attempts=3,
        context_length=8192,
        return_step_meta=False,
        output_type="str",
        auto_generate_prompt=False,
```

```
        max_tokens=4000,
        stopping_token="<DONE>",
        interactive=False,
    )
    logger.success("Agent initialized successfully")
    return agent


# Step 5: Run the Agent with Data
def summarize_commits_with_agent(agent: Agent, prompt: str) -> str:
    """
    Pass the system prompt to the agent and fetch the result.
    """
    logger.info("Sending data to the agent for summarization")
    try:
        result = agent.run(
            f"{prompt}",
            all_cores=True,
        )
        logger.success("Agent completed the summarization task")
        return result
    except Exception as e:
        logger.error(f"Agent encountered an error: {e}")
        raise
```

```
# Main Execution
if __name__ == "__main__":
    try:
        logger.info("Starting commit summarization process")

        # Fetch latest commits
        latest_commits = fetch_latest_commits(GITHUB_API_URL, limit=5)

        # Format commits and get current time
        current_time, commit_summary = format_commits_with_time(
            latest_commits
        )

        # Build the custom system prompt
        custom_system_prompt = build_custom_system_prompt(
            current_time, commit_summary
        )

        # Initialize agent
        agent = initialize_agent()

        # Set the dynamic system prompt
        agent.system_prompt = custom_system_prompt

        # Run the agent and summarize commits
        result = summarize_commits_with_agent(
```

```
        agent, custom_system_prompt
    )

    # Print the result
    print("### Commit Summary in Markdown:")
    print(result)

except Exception as e:
    logger.critical(f"Process failed: {e}")
```