Swarms 5.8.1 Feature Documentation

1. Enhanced Command Line Interface (CLI)

1.1 Integrated Onboarding Process

```bash
$ swarms onboarding
```

1.2 Run Agents Command

```bash
$ swarms run-agents --yaml-file agents.yaml
```

This command allows users to execute multiple agents defined in a YAML file. Here's the process:

1. The command reads the specified YAML file (`agents.yaml` in this case).
2. It parses the YAML content, extracting the configuration for each agent.
3. For each agent defined in the YAML:
   - It creates an instance of the agent with the specified parameters.
   - It sets up the agent's environment (model, temperature, max tokens, etc.).
   - It assigns the given task to the agent.
   - It executes the agent, respecting parameters like `max_loops`, `autosave`, and `verbose`.
4. The results from all agents are collected and presented to the user.

The YAML file structure allows users to define multiple agents with different configurations, making it easy to run complex, multi-agent tasks from the command line.

1.3 Generate Prompt Feature

```bash
$ swarms generate-prompt --prompt "Create a marketing strategy for a new product launch"
```

This feature leverages Swarms' language model to generate expanded or refined prompts:

1. The command takes the user's input prompt as a starting point.
2. It likely sends this prompt to a pre-configured language model (possibly GPT-4 or a similar model).
3. The model then generates an expanded or more detailed version of the prompt.
4. The generated prompt is returned to the user, possibly with options to further refine or save it.

This feature can help users create more effective prompts for their agents or other AI tasks.

2. New Prompt Management System

2.1 Prompt Class

The new `Prompt` class provides a robust system for managing and versioning prompts:

```python
from swarms import Prompt

marketing_prompt = Prompt(content="Initial marketing strategy draft", autosave=True)

print(marketing_prompt.get_prompt())
```

Key features of the `Prompt` class:

1. **Initialization**: The class is initialized with initial content and an `autosave` option.

2. **Editing**:
   ```python
   marketing_prompt.edit_prompt("Updated marketing strategy with social media focus")
   ```
   This method updates the prompt content and, if `autosave` is True, automatically saves the new version.

3. **Retrieval**:
   ```python
   current_content = marketing_prompt.get_prompt()
   ```
   This method returns the current content of the prompt.

4. **Version History**:

```python
print(f"Edit history: {marketing_prompt.edit_history}")
```

The class maintains a history of edits, allowing users to track changes over time.

5. **Rollback**:

```python
marketing_prompt.rollback(1)
```

This feature allows users to revert to a previous version of the prompt.

6. **Duplicate Prevention**:

The class includes logic to prevent duplicate edits, raising a `ValueError` if an attempt is made to save the same content twice in a row.

This system provides a powerful way to manage prompts, especially for complex projects where prompt engineering and iteration are crucial.

3. Upcoming Features Preview

3.1 Enhanced Agent Execution Capabilities

The preview code demonstrates planned enhancements for agent execution:

```python
from swarms import Agent, ExecutionEnvironment
```

```
my_agent = Agent(name="data_processor")

cpu_env = ExecutionEnvironment(type="cpu", cores=4)
my_agent.run(environment=cpu_env)

gpu_env = ExecutionEnvironment(type="gpu", device_id=0)
my_agent.run(environment=gpu_env)

fractional_env = ExecutionEnvironment(type="fractional", cpu_fraction=0.5, gpu_fraction=0.3)
my_agent.run(environment=fractional_env)
```

This upcoming feature will allow for more fine-grained control over the execution environment:

1. **CPU Execution**: Users can specify the number of CPU cores to use.
2. **GPU Execution**: Allows selection of a specific GPU device.
3. **Fractionalized Execution**: Enables partial allocation of CPU and GPU resources.

These features will provide users with greater flexibility in resource allocation, potentially improving performance and allowing for more efficient use of available hardware.