```
import os
from swarms import Agent
from swarm_models import OpenAIChat
from dotenv import load_dotenv

# Custom system prompt for VC legal document generation
VC_LEGAL_AGENT_PROMPT = """You are a specialized legal document assistant focusing on venture capital documentation.
Your role is to help draft preliminary versions of common VC legal documents while adhering to these guidelines:

1. Always include standard legal disclaimers
2. Follow standard VC document structures
3. Flag areas that need attorney review
4. Request necessary information for document completion
5. Maintain consistency across related documents
6. Output <DONE> only when document is complete and verified

Remember: All output should be marked as 'DRAFT' and require professional legal review."""


def create_vc_legal_agent():
    load_dotenv()
    api_key = os.getenv("OPENAI_API_KEY")

    # Configure the model with appropriate parameters for legal work
```

```
# Get the OpenAI API key from the environment variable
api_key = os.getenv("GROQ_API_KEY")


# Model
model = OpenAIChat(
    openai_api_base="https://api.groq.com/openai/v1",
    openai_api_key=api_key,
    model_name="llama-3.1-70b-versatile",
    temperature=0.1,
)


# Initialize the persistent agent
agent = Agent(
    agent_name="VC-Legal-Document-Agent",
    system_prompt=VC_LEGAL_AGENT_PROMPT,
    llm=model,
    max_loops="auto",  # Allows multiple iterations until completion
    stopping_token="<DONE>",  # Agent will continue until this token is output
    autosave=True,
    dashboard=True,  # Enable dashboard for monitoring
    verbose=True,
    dynamic_temperature_enabled=False,  # Disable for consistency in legal documents
    saved_state_path="vc_legal_agent_state.json",
    user_name="legal_corp",
    retry_attempts=3,
    context_length=200000,
```

```
        return_step_meta=True,
        output_type="string",
        streaming_on=False,
    )

    return agent


def generate_legal_document(agent, document_type, parameters):
    """
    Generate a legal document with multiple refinement iterations

    Args:
        agent: The initialized VC legal agent
        document_type: Type of document to generate (e.g., "term_sheet", "investment_agreement")
        parameters: Dict containing necessary parameters for the document

    Returns:
        str: The generated document content
    """
    prompt = f"""
    Generate a {document_type} with the following parameters:
    {parameters}

    Please follow these steps:
    1. Create initial draft
```

```
    2. Review for completeness
    3. Add necessary legal disclaimers
    4. Verify all required sections
    5. Output <DONE> when complete

    Include [REQUIRES LEGAL REVIEW] tags for sections needing attorney attention.
    """

    return agent.run(prompt)


# Example usage
if __name__ == "__main__":
    # Initialize the agent
    legal_agent = create_vc_legal_agent()

    # Example parameters for a term sheet
    parameters = {
        "company_name": "TechStartup Inc.",
        "investment_amount": "$5,000,000",
        "valuation": "$20,000,000",
        "investor_rights": [
            "Board seat",
            "Pro-rata rights",
            "Information rights",
        ],
```

```
    "type_of_security": "Series A Preferred Stock",
}

# Generate a term sheet
document = generate_legal_document(
    legal_agent, "term_sheet", parameters
)

# Save the generated document
with open("generated_term_sheet_draft.md", "w") as f:
    f.write(document)
```