```
import pytest
from unittest.mock import Mock, patch
from swarms.structs.mixture_of_agents import MixtureOfAgents
from swarms.structs.agent import Agent
from swarms_memory import BaseVectorDatabase


def test_init():
    with patch.object(
        MixtureOfAgents, "agent_check"
    ) as mock_agent_check, patch.object(
        MixtureOfAgents, "final_agent_check"
    ) as mock_final_agent_check, patch.object(
        MixtureOfAgents, "swarm_initialization"
    ) as mock_swarm_initialization, patch.object(
        MixtureOfAgents, "communication_protocol"
    ) as mock_communication_protocol:
        agents = [Mock(spec=Agent)]
        final_agent = Mock(spec=Agent)
        scp = Mock(spec=BaseVectorDatabase)
        MixtureOfAgents(
            agents=agents, final_agent=final_agent, scp=scp
        )
        mock_agent_check.assert_called_once()
        mock_final_agent_check.assert_called_once()
        mock_swarm_initialization.assert_called_once()
```

```
        mock_communication_protocol.assert_called_once()


def test_communication_protocol():
    agents = [Mock(spec=Agent)]
    final_agent = Mock(spec=Agent)
    scp = Mock(spec=BaseVectorDatabase)
    swarm = MixtureOfAgents(
        agents=agents, final_agent=final_agent, scp=scp
    )
    swarm.communication_protocol()
    for agent in agents:
        agent.long_term_memory.assert_called_once_with(scp)


def test_agent_check():
    final_agent = Mock(spec=Agent)
    with pytest.raises(TypeError):
        MixtureOfAgents(agents="not a list", final_agent=final_agent)
    with pytest.raises(TypeError):
        MixtureOfAgents(
            agents=["not an agent"], final_agent=final_agent
        )


def test_final_agent_check():
```

```
    agents = [Mock(spec=Agent)]
    with pytest.raises(TypeError):
        MixtureOfAgents(agents=agents, final_agent="not an agent")


def test_swarm_initialization():
    with patch(
        "swarms.structs.mixture_of_agents.logger"
    ) as mock_logger:
        agents = [Mock(spec=Agent)]
        final_agent = Mock(spec=Agent)
        swarm = MixtureOfAgents(
            agents=agents, final_agent=final_agent
        )
        swarm.swarm_initialization()
        assert mock_logger.info.call_count == 3


def test_run():
    with patch("swarms.structs.mixture_of_agents.logger"), patch(
        "builtins.open", new_callable=Mock
    ) as mock_open:
        agents = [Mock(spec=Agent)]
        final_agent = Mock(spec=Agent)
        swarm = MixtureOfAgents(
            agents=agents, final_agent=final_agent
```

```
)

swarm.run("task")

for agent in agents:

    agent.run.assert_called_once()

final_agent.run.assert_called_once()

mock_open.assert_called_once_with(swarm.saved_file_name, "w")
```