```
from unittest.mock import MagicMock

import pytest

from swarms.structs.agent import Agent
from swarms.structs.majority_voting import MajorityVoting


def test_majority_voting_run_concurrent(mocker):
    # Create mock agents
    agent1 = MagicMock(spec=Agent)
    agent2 = MagicMock(spec=Agent)
    agent3 = MagicMock(spec=Agent)

    # Create mock majority voting
    mv = MajorityVoting(
        agents=[agent1, agent2, agent3],
        concurrent=True,
        multithreaded=False,
    )

    # Create mock conversation
    conversation = MagicMock()
    mv.conversation = conversation

    # Create mock results
```

```
results = ["Paris", "Paris", "Lyon"]

# Mock agent.run method
agent1.run.return_value = results[0]
agent2.run.return_value = results[1]
agent3.run.return_value = results[2]

# Run majority voting
majority_vote = mv.run("What is the capital of France?")

# Assert agent.run method was called with the correct task
agent1.run.assert_called_once_with(
    "What is the capital of France?"
)
agent2.run.assert_called_once_with(
    "What is the capital of France?"
)
agent3.run.assert_called_once_with(
    "What is the capital of France?"
)

# Assert conversation.add method was called with the correct responses
conversation.add.assert_any_call(agent1.agent_name, results[0])
conversation.add.assert_any_call(agent2.agent_name, results[1])
conversation.add.assert_any_call(agent3.agent_name, results[2])
```

```
    # Assert majority vote is correct
    assert majority_vote is not None


def test_majority_voting_run_multithreaded(mocker):
    # Create mock agents
    agent1 = MagicMock(spec=Agent)
    agent2 = MagicMock(spec=Agent)
    agent3 = MagicMock(spec=Agent)

    # Create mock majority voting
    mv = MajorityVoting(
        agents=[agent1, agent2, agent3],
        concurrent=False,
        multithreaded=True,
    )

    # Create mock conversation
    conversation = MagicMock()
    mv.conversation = conversation

    # Create mock results
    results = ["Paris", "Paris", "Lyon"]

    # Mock agent.run method
    agent1.run.return_value = results[0]
```

```
agent2.run.return_value = results[1]
agent3.run.return_value = results[2]

# Run majority voting
majority_vote = mv.run("What is the capital of France?")

# Assert agent.run method was called with the correct task
agent1.run.assert_called_once_with(
    "What is the capital of France?"
)
agent2.run.assert_called_once_with(
    "What is the capital of France?"
)
agent3.run.assert_called_once_with(
    "What is the capital of France?"
)

# Assert conversation.add method was called with the correct responses
conversation.add.assert_any_call(agent1.agent_name, results[0])
conversation.add.assert_any_call(agent2.agent_name, results[1])
conversation.add.assert_any_call(agent3.agent_name, results[2])

# Assert majority vote is correct
assert majority_vote is not None
```

```
@pytest.mark.asyncio
async def test_majority_voting_run_asynchronous(mocker):
    # Create mock agents
    agent1 = MagicMock(spec=Agent)
    agent2 = MagicMock(spec=Agent)
    agent3 = MagicMock(spec=Agent)

    # Create mock majority voting
    mv = MajorityVoting(
        agents=[agent1, agent2, agent3],
        concurrent=False,
        multithreaded=False,
        asynchronous=True,
    )

    # Create mock conversation
    conversation = MagicMock()
    mv.conversation = conversation

    # Create mock results
    results = ["Paris", "Paris", "Lyon"]

    # Mock agent.run method
    agent1.run.return_value = results[0]
    agent2.run.return_value = results[1]
    agent3.run.return_value = results[2]
```

```
# Run majority voting
majority_vote = await mv.run("What is the capital of France?")

# Assert agent.run method was called with the correct task
agent1.run.assert_called_once_with(
    "What is the capital of France?"
)
agent2.run.assert_called_once_with(
    "What is the capital of France?"
)
agent3.run.assert_called_once_with(
    "What is the capital of France?"
)

# Assert conversation.add method was called with the correct responses
conversation.add.assert_any_call(agent1.agent_name, results[0])
conversation.add.assert_any_call(agent2.agent_name, results[1])
conversation.add.assert_any_call(agent3.agent_name, results[2])

# Assert majority vote is correct
assert majority_vote is not None
```