```
import os
from datetime import datetime

import pytest

from swarms.structs.base_structure import BaseStructure


class TestBaseStructure:
    def test_init(self):
        base_structure = BaseStructure(
            name="TestStructure",
            description="Test description",
            save_metadata=True,
            save_artifact_path="./test_artifacts",
            save_metadata_path="./test_metadata",
            save_error_path="./test_errors",
        )

        assert base_structure.name == "TestStructure"
        assert base_structure.description == "Test description"
        assert base_structure.save_metadata is True
        assert base_structure.save_artifact_path == "./test_artifacts"
        assert base_structure.save_metadata_path == "./test_metadata"
        assert base_structure.save_error_path == "./test_errors"
```

```
def test_save_to_file_and_load_from_file(self, tmpdir):
    tmp_dir = tmpdir.mkdir("test_dir")
    file_path = os.path.join(tmp_dir, "test_file.json")

    data_to_save = {"key": "value"}
    base_structure = BaseStructure()

    base_structure.save_to_file(data_to_save, file_path)
    loaded_data = base_structure.load_from_file(file_path)

    assert loaded_data == data_to_save

def test_save_metadata_and_load_metadata(self, tmpdir):
    tmp_dir = tmpdir.mkdir("test_dir")
    base_structure = BaseStructure(save_metadata_path=tmp_dir)

    metadata = {"name": "Test", "description": "Test metadata"}
    base_structure.save_metadata(metadata)
    loaded_metadata = base_structure.load_metadata()

    assert loaded_metadata == metadata

def test_log_error(self, tmpdir):
    tmp_dir = tmpdir.mkdir("test_dir")
    base_structure = BaseStructure(save_error_path=tmp_dir)
```

```
    error_message = "Test error message"
    base_structure.log_error(error_message)

    log_file = os.path.join(tmp_dir, "TestStructure_errors.log")
    with open(log_file) as file:
        lines = file.readlines()
        assert len(lines) == 1
        assert lines[0] == f"{error_message}\n"

def test_save_artifact_and_load_artifact(self, tmpdir):
    tmp_dir = tmpdir.mkdir("test_dir")
    base_structure = BaseStructure(save_artifact_path=tmp_dir)

    artifact = {"key": "value"}
    artifact_name = "test_artifact"
    base_structure.save_artifact(artifact, artifact_name)
    loaded_artifact = base_structure.load_artifact(artifact_name)

    assert loaded_artifact == artifact

def test_current_timestamp(self):
    base_structure = BaseStructure()
    current_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    timestamp = base_structure._current_timestamp()
    assert timestamp == current_time
```

```
def test_log_event(self, tmpdir):
    tmp_dir = tmpdir.mkdir("test_dir")
    base_structure = BaseStructure(save_metadata_path=tmp_dir)

    event = "Test event"
    event_type = "INFO"
    base_structure.log_event(event, event_type)

    log_file = os.path.join(tmp_dir, "TestStructure_events.log")
    with open(log_file) as file:
        lines = file.readlines()
        assert len(lines) == 1
        assert (
            lines[0] == f"[{base_structure._current_timestamp()}]"
            f" [{event_type}] {event}\n"
        )

@pytest.mark.asyncio
async def test_run_async(self):
    base_structure = BaseStructure()

    async def async_function():
        return "Async Test Result"

    result = await base_structure.run_async(async_function)
    assert result == "Async Test Result"
```

```
@pytest.mark.asyncio
async def test_save_metadata_async(self, tmpdir):
    tmp_dir = tmpdir.mkdir("test_dir")
    base_structure = BaseStructure(save_metadata_path=tmp_dir)

    metadata = {"name": "Test", "description": "Test metadata"}
    await base_structure.save_metadata_async(metadata)
    loaded_metadata = base_structure.load_metadata()

    assert loaded_metadata == metadata

@pytest.mark.asyncio
async def test_log_error_async(self, tmpdir):
    tmp_dir = tmpdir.mkdir("test_dir")
    base_structure = BaseStructure(save_error_path=tmp_dir)

    error_message = "Test error message"
    await base_structure.log_error_async(error_message)

    log_file = os.path.join(tmp_dir, "TestStructure_errors.log")
    with open(log_file) as file:
        lines = file.readlines()
        assert len(lines) == 1
        assert lines[0] == f"{error_message}\n"
```

```
@pytest.mark.asyncio
async def test_save_artifact_async(self, tmpdir):
    tmp_dir = tmpdir.mkdir("test_dir")
    base_structure = BaseStructure(save_artifact_path=tmp_dir)

    artifact = {"key": "value"}
    artifact_name = "test_artifact"
    await base_structure.save_artifact_async(
        artifact, artifact_name
    )
    loaded_artifact = base_structure.load_artifact(artifact_name)

    assert loaded_artifact == artifact

@pytest.mark.asyncio
async def test_load_artifact_async(self, tmpdir):
    tmp_dir = tmpdir.mkdir("test_dir")
    base_structure = BaseStructure(save_artifact_path=tmp_dir)

    artifact = {"key": "value"}
    artifact_name = "test_artifact"
    base_structure.save_artifact(artifact, artifact_name)
    loaded_artifact = await base_structure.load_artifact_async(
        artifact_name
    )
```

```
    assert loaded_artifact == artifact


@pytest.mark.asyncio
async def test_log_event_async(self, tmpdir):
    tmp_dir = tmpdir.mkdir("test_dir")
    base_structure = BaseStructure(save_metadata_path=tmp_dir)

    event = "Test event"
    event_type = "INFO"
    await base_structure.log_event_async(event, event_type)

    log_file = os.path.join(tmp_dir, "TestStructure_events.log")
    with open(log_file) as file:
        lines = file.readlines()
        assert len(lines) == 1
        assert (
            lines[0] == f"[{base_structure._current_timestamp()}]"
            f" [{event_type}] {event}\n"
        )


@pytest.mark.asyncio
async def test_asave_to_file(self, tmpdir):
    tmp_dir = tmpdir.mkdir("test_dir")
    file_path = os.path.join(tmp_dir, "test_file.json")
    data_to_save = {"key": "value"}
    base_structure = BaseStructure()
```

```
    await base_structure.asave_to_file(data_to_save, file_path)
    loaded_data = base_structure.load_from_file(file_path)

    assert loaded_data == data_to_save

@pytest.mark.asyncio
async def test_aload_from_file(self, tmpdir):
    tmp_dir = tmpdir.mkdir("test_dir")
    file_path = os.path.join(tmp_dir, "test_file.json")
    data_to_save = {"key": "value"}
    base_structure = BaseStructure()
    base_structure.save_to_file(data_to_save, file_path)

    loaded_data = await base_structure.aload_from_file(file_path)
    assert loaded_data == data_to_save

def test_run_in_thread(self):
    base_structure = BaseStructure()
    result = base_structure.run_in_thread(
        lambda: "Thread Test Result"
    )
    assert result.result() == "Thread Test Result"

def test_save_and_decompress_data(self):
    base_structure = BaseStructure()
```

```
    data = {"key": "value"}
    compressed_data = base_structure.compress_data(data)
    decompressed_data = base_structure.decompres_data(
        compressed_data
    )
    assert decompressed_data == data

def test_run_batched(self):
    base_structure = BaseStructure()

    def run_function(data):
        return f"Processed {data}"

    batched_data = list(range(10))
    result = base_structure.run_batched(
        batched_data, batch_size=5, func=run_function
    )

    expected_result = [
        f"Processed {data}" for data in batched_data
    ]
    assert result == expected_result

def test_load_config(self, tmpdir):
    tmp_dir = tmpdir.mkdir("test_dir")
    config_file = os.path.join(tmp_dir, "config.json")
```

```
    config_data = {"key": "value"}
    base_structure = BaseStructure()

    base_structure.save_to_file(config_data, config_file)
    loaded_config = base_structure.load_config(config_file)

    assert loaded_config == config_data

def test_backup_data(self, tmpdir):
    tmp_dir = tmpdir.mkdir("test_dir")
    base_structure = BaseStructure()
    data_to_backup = {"key": "value"}
    base_structure.backup_data(
        data_to_backup, backup_path=tmp_dir
    )
    backup_files = os.listdir(tmp_dir)

    assert len(backup_files) == 1
    loaded_data = base_structure.load_from_file(
        os.path.join(tmp_dir, backup_files[0])
    )
    assert loaded_data == data_to_backup

def test_monitor_resources(self):
    base_structure = BaseStructure()
    base_structure.monitor_resources()
```

```
def test_run_with_resources(self):
    base_structure = BaseStructure()

    def run_function():
        base_structure.monitor_resources()
        return "Resource Test Result"

    result = base_structure.run_with_resources(run_function)
    assert result == "Resource Test Result"

def test_run_with_resources_batched(self):
    base_structure = BaseStructure()

    def run_function(data):
        base_structure.monitor_resources()
        return f"Processed {data}"

    batched_data = list(range(10))
    result = base_structure.run_with_resources_batched(
        batched_data, batch_size=5, func=run_function
    )

    expected_result = [
        f"Processed {data}" for data in batched_data
    ]
```

```
assert result == expected_result
```