**Name : Welly**

**E-mail :** [**wellytan09@gmail.com**](mailto:wellytan09@gmail.com)

using Newtonsoft.Json;

using NUnit.Framework;

using System;

namespace RepositoryManager.Tests

{

[TestFixture]

public class Class1Tests

{

private Class1 \_repository;

[SetUp]

public void Setup()

{

\_repository = new Class1();

}

// Test Case 1: Initialization

[Test]

public void Initialization\_ShouldCompleteWithoutException()

{

Assert.DoesNotThrow(() => new Class1());

}

// Test Case 2: Register Method

[Test]

public void Register\_ShouldAddItemSuccessfully()

{

Assert.DoesNotThrow(() => \_repository.Register("item1", "{\"key\": \"value\"}", 1));

Assert.AreEqual("{\"key\": \"value\"}", \_repository.Retrieve("item1"));

}

[Test]

public void Register\_ShouldThrowExceptionForDuplicateItem()

{

\_repository.Register("item1", "{\"key\": \"value\"}", 1);

var ex = Assert.Throws<InvalidOperationException>(() => \_repository.Register("item1", "{\"key\": \"value\"}", 1));

Assert.AreEqual("Item with name 'item1' already exists.", ex.Message);

}

// Test Case 3: Retrieve Method

[Test]

public void Retrieve\_ShouldReturnItemContent()

{

\_repository.Register("item1", "{\"key\": \"value\"}", 1);

var content = \_repository.Retrieve("item1");

Assert.AreEqual("{\"key\": \"value\"}", content);

}

[Test]

public void Retrieve\_ShouldThrowExceptionForNonExistentItem()

{

var ex = Assert.Throws<KeyNotFoundException>(() => \_repository.Retrieve("nonexistentItem"));

Assert.AreEqual("Item with name 'nonexistentItem' not found.", ex.Message);

}

// Test Case 4: GetType Method

[Test]

public void GetType\_ShouldReturn1ForValidJSON()

{

var type = \_repository.GetType("{\"key\": \"value\"}");

Assert.AreEqual(1, type);

}

[Test]

public void GetType\_ShouldReturn2ForValidXML()

{

var type = \_repository.GetType("<root><key>value</key></root>");

Assert.AreEqual(2, type);

}

[Test]

public void GetType\_ShouldThrowExceptionForInvalidContent()

{

var ex = Assert.Throws<InvalidDataException>(() => \_repository.GetType("plain text"));

Assert.AreEqual("Item content is neither JSON nor XML.", ex.Message);

}

// Test Case 5: Deregister Method

[Test]

public void Deregister\_ShouldRemoveItemSuccessfully()

{

\_repository.Register("item1", "{\"key\": \"value\"}", 1);

Assert.DoesNotThrow(() => \_repository.Deregister("item1"));

}

[Test]

public void Deregister\_ShouldThrowExceptionForNonExistentItem()

{

var ex = Assert.Throws<KeyNotFoundException>(() => \_repository.Deregister("nonexistentItem"));

Assert.AreEqual("Item with name 'nonexistentItem' not found or could not be removed.", ex.Message);

}

// Test Case 6: Validate Method

[Test]

public void Validate\_ShouldPassForValidJSON()

{

Assert.DoesNotThrow(() => \_repository.Validate("{\"key\": \"value\"}", 1));

}

[Test]

public void Validate\_ShouldThrowExceptionForInvalidJSON()

{

var ex = Assert.Throws<FormatException>(() => \_repository.Validate("{key: value}", 1));

Assert.AreEqual("Invalid JSON format.", ex.Message);

}

[Test]

public void Validate\_ShouldPassForValidXML()

{

Assert.DoesNotThrow(() => \_repository.Validate("<root><key>value</key></root>", 2));

}

[Test]

public void Validate\_ShouldThrowExceptionForInvalidXML()

{

var ex = Assert.Throws<FormatException>(() => \_repository.Validate("<root><key>value</key>", 2));

Assert.AreEqual("Invalid XML format.", ex.Message);

}

}

}

<https://github.com/DoomedMean/Company-Test/tree/main/Formatrix>