Name : Welly
E-mail : 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