**Logo

Description automatically generated**

Testing Framework for C# and .NET

Report Subtitle Goes Here

@Copyright 2021 Version 1 – All Rights Reserved

Company Classification: Internal

Contents

[1. User Guide 4](#_Toc85374493)

[1.1. Test Automation Guidelines 4](#_Toc85374494)

[1.2. Test planning 4](#_Toc85374495)

[1.3. The Framework 4](#_Toc85374496)

[2. Installation 6](#_Toc85374497)

[3. Using the FRamework 7](#_Toc85374498)

[3.1. BDD – 7](#_Toc85374499)

[4. Sample stuff from Template 8](#_Toc85374500)

[4.1. Pie Chart 8](#_Toc85374501)

[4.2. Bar Chart Title 8](#_Toc85374502)

[4.3. Smart Art Graphic 1 9](#_Toc85374503)

[4.4. Smart Art Graphic 2 9](#_Toc85374504)

Version Control

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Changed By | Reason for Change |
|  |  |  |  |
|  |  |  |  |

Circulation List

|  |  |
| --- | --- |
| Name | Organisation/Title |
|  |  |
|  |  |

Reference Documents

|  |  |  |
| --- | --- | --- |
| Title | Description | Owner |
|  |  |  |
|  |  |  |

# **User Guide**

## Test Automation Guidelines

Automation

## Test planning

Planning

### What to Test

A picture containing chart

Description automatically generated

UI testing is expensive, time consuming and usually brittle, it may fail at random times and perform satisfactorily most of the time. Therefore the better testing frameworks limit UI testing to what cannot be tested at a lower level, for instance validating a REST api or service. These in turn an be simplified if there is good unit tests and data validation – for example, testing database procedures, validating file imports etc.

## Where to test

In a cloud based, agile environment, the proper approach is to perform as much testing as possible each time a PR is submitted. That ensures that new or modified code is tested before being allowed to be added to a build.

## When to Test

Shift Left – test early and test often so as to know as soon as possible that an error condition may exists

Shift Right – testing at the end of development and avoiding many man-hours of finding failures and curtail many iterations of testing.

## Coding Standards

All C# code should follow the conventions recommended by Microsoft at:

[C# Coding Conventions | Microsoft Docs](https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/coding-conventions)

[Naming Guidelines - Framework Design Guidelines | Microsoft Docs](https://docs.microsoft.com/en-us/dotnet/standard/design-guidelines/naming-guidelines)

Thread.sleep should not be used at any time as it stops processing.

Selenium Implicit waits should only be used when the browser response is guaranteed to be always consistent.

Selenium Implicit waits should not be mixed with Explicit or Fluent waits.

The preferred locator for web elements is the ID as it is least likely to be changed when a page is modified, many changes are presentational using changed CSS.

## The Framework

### Logging

Logging in the framework is important, it is the answer to investigating the cause of a test failing in the application. The cause could be:

#### A development fault – Code has changed and does not deliver the expected results.

#### A testing fault – a test no longer reflects how the application works.

#### An environment fault – the application and the tests work as expected but there still are faults to be investigated.

Good logging clarifies the cause of any fault. The best logging describes what was expected and what caused the failure:

“Expected dropdown list – Affected County – to contain Yorkshire, actual values are [Powys, Glamorgan]”

“Expected Interest Amount to be £3.44 but was £299 when given a deposit of £540”

More detail should be included if this is too ambiguous such as these fields appearing on multiple pages then the page name should be included in the error.

### Reporting

### Performance

In this context, performance is a measure of testing over the life of the automation, it should be used to identify tests that frequently break, regular metrics on how automation has worked etc.

# Installation

# Using the FRamework

## BDD –

Behaviour Driven Development is an

### Guidelines

* Feature files should have few scenarios, with most tools, multiple features can run in parallel but scenarios do not.
* A scenario should be a complete test, it should not depend on any other scenario. Therefore if it is necessary to import data then that should be part of the GIVEN
* A scenario should be granular and have only one test. For example “WHEN I click on ‘Add to Basket’” should be the only action for a WHEN statement. GIVEN should be used to perform what actions are required to reach that point.

# Sample stuff from Template

## Pie Chart

Figure

## Bar Chart Title

Figure

## Smart Art Graphic 1

Figure

## Smart Art Graphic 2

Figure

Figure

A black and white logo

Description automatically generated with low confidenceShape

Description automatically generatedA picture containing person, indoor

Description automatically generated

**Thank you**

**Contact:**

w: [version1.com](https://www.version1.com/)

e: [info@version1.com](mailto:info@version1.com)