# CA1 – Lab 2

## Introduction:

This lab is aimed at exploring test frameworks and supporting software as part of a DevOps pipeline. It will explore options to incorporate testing frameworks into the DevOps pipeline using industry standard tooling and principles focused on delivering quicker feedback in the software development lifecycle with the end goal of producing better quality software more efficiently than older software development lifecycle models. It will illustrate the use of the tools and also a high level conclusion on the system as a whole and the value it delivers under DevOps as well as a personal reflection on the learnings and discoveries made in completing the lab.

## Aims/Objectives:

* Research and explore available solutions for testing in the DevOps pipeline
* Choose a suitable tool
* Illustrate the use of this tool in a project
* Demonstrate the value produced by the selected tool based on reports generated from the system with the tool incorporated.
* Explore the concept of testing in the DevOps and Software Development Lifecycle.
* Gain a better understanding of the different types of testing tools and where they fit in at the various stages of software development and the DevOps pipeline and the value they bring to each stage.
* Draw up a conclusion on why it is important to incorporate these ideas into the pipeline and the benefit they have over not having it as well a comparison to using older testing methodologies.
* Explore the concept of “shifting left” in regards to testing and illustrate this understanding in the conclusion

## Method:

* Researched testing frameworks in DevOps and categorised different tools for different areas and stages of the pipeline
* Decided on a language and framework that would be used as a pre-requisite to choosing a testing framework.
* Chose two tools that could be used at various stages of the pipeline, where one would be integrated into the pipeline and another would be used as a comparison to explore the differences in available frameworks and where each one would fit into a pipeline.
* Created a repository to host the solution that would be used to integrate the chosen testing framework.
* Created a C# ASP.Net Web App using the default Visual Studio 2022 template targeting .Net 7
* Followed Microsoft Learn tutorial cited in conclusion to build up a working minimal API to be used for testing.
* Downloaded Postman desktop app to use for testing API.
* Created a collection in Postman to store tests using different HTTP methods on the developed API
* Ran Postman tests on the API as the solution was being developed further

## Results:

## Conclusion:

## References:

[Tutorial: Create a minimal web API with ASP.NET Core | Microsoft Learn](https://learn.microsoft.com/en-us/aspnet/core/tutorials/min-web-api?view=aspnetcore-7.0&tabs=visual-studio)

[How to unit test Minimal APIs in .NET 6 (and why it's hard) - YouTube](https://www.youtube.com/watch?v=VuFQtyRmS0E)

[Microsoft fixed my biggest complaint with Minimal APIs in .NET 7 - YouTube](https://www.youtube.com/watch?v=-i4rP0LGY5U)

[Test Minimal API apps | Microsoft Learn](https://learn.microsoft.com/en-us/aspnet/core/fundamentals/minimal-apis/test-min-api?view=aspnetcore-7.0)

[AspNetCore.Docs.Samples/fundamentals/minimal-apis/samples/MinApiTestsSample at main · dotnet/AspNetCore.Docs.Samples (github.com)](https://github.com/dotnet/AspNetCore.Docs.Samples/tree/main/fundamentals/minimal-apis/samples/MinApiTestsSample)