*DOWNHILL*

TESTING PLAN

Version *1.0*

*02/03/2021*

TABLE OF CONTENTS

[Testing Procedure](#_heading=h.3znysh7) **1**

[1.1 Unit Testing](#_heading=h.2et92p0) 1

[1.2 Test Execution on Pull Requests](#_heading=h.tyjcwt) 2

[1.3 Code coverage](#_heading=h.3dy6vkm) 2

[2. Weekly activities](#_heading=h.1t3h5sf) **3**

[2.1 Weekly Tests](#_heading=h.omqf9ms22wlg) 3

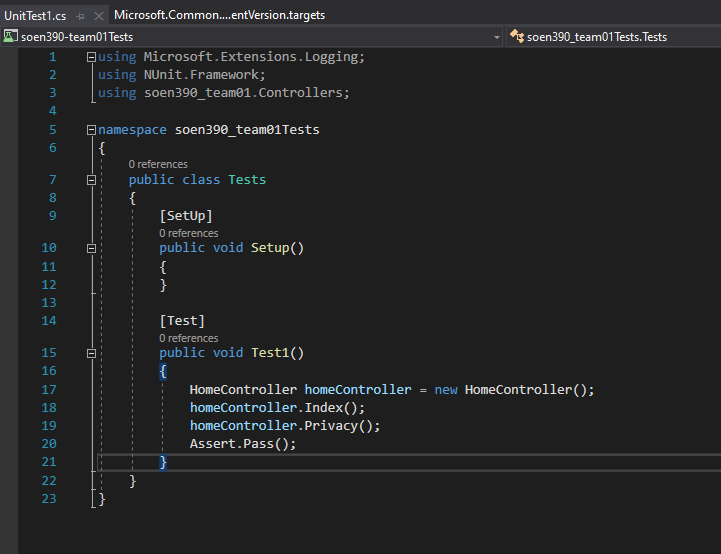
# 

# Testing Procedure

## 1.1 Unit Testing

We will be using NUnit as our testing framework for our Unit tests. We chose to use NUnit since it is widely considered as one of the best unit-testing frameworks for .NET languages and a few members of our team have experience with it. We will be using version 3.13 of NUnit since it is the latest and most up to date version. It can be installed from the NUnit github page at the following link: <https://github.com/nunit/nunit/releases/tag/v3.13>

We have already configured this into our code and have a dummy test set up in the file “UnitTest1.cs” as seen below:



Another reason we chose to use NUnit is due to the suite attribute which allows for aggregating of tests to execute separately which is useful in projects like ours.

## 1.2 Test Execution on Pull Requests

We will be using SonarCloud for code analysis and to detect quality issues upon pull requests being made. SonarCloud uses the static code analysis approach to allow for easrly detection of issues in order to increase the overall quality of the code. We already have SonarCloud configured with our repository on GitHub.

There are 3 kinds of issues that SonarCloud detects:

Code Smells: These are characteristic of the code that, while not actually preventing the proper functioning of the program, may indicate deeper problems that negatively affect the maintainability of the code. Early identification of these types of issues can help to alleviate technical debt in the application.

Bugs: These are errors in the code that can prevent the program from operating as intended. They affect code reliability.

Vulnerabilities: These are problems in the code that could be exploited by a bad actor to compromise the security of the application.

As mentioned previously, this is already configured on our GitHub repository and can be verified.

### 1.3 Code coverage

We will be using the Coverlet library for code coverage.

Coverlet is an open source project on GitHub that provides a cross platform code coverage framework for C#. Coverlet is part of the .NET foundation. The following document details how to configure Coverlet: <https://docs.microsoft.com/en-us/dotnet/core/testing/unit-testing-code-coverage?tabs=windows> .

# 2. Weekly activities

## 2.1 Weekly Tests

1. Build and run the most up to date version of the project. A specific set of activities should be performed in order to maintain consistency. For the same reason, the same machine should be used to perform the weekly tests.
2. Weekly review of bugs:

* Verify that “fixed” bugs are really fixed and do not persist.
* Rank bugs relative to their urgency as well as the progress of the project as a whole.
* Generate a weekly report of fixed bugs.