CitySim2017 Findings Report

Created By: Jack Waayer

# Introduction

This document details the test findings for the CitySim2017 console application. Discoveries, difficulties and observations are discussed along with reflection of the testing process.

Unit testing was the sole testing approach used to test the CitySim2017 application. During the testing process we were developing our skills in the Visual Studio 2015 environment using the C# language. Along the way there were many difficulties and discoveries which we encountered.

# Findings

The Requirements for the CitySim2017 application were challenging to understand, but after analysing the different functions and making some assumptions the aim became clear. After we had interpreted the requirements we started development on the test plans. These proved to be difficult, needing a large amount of time to develop.

Once the planning was complete we started development of the unit tests. The .NET framework includes its own unit testing framework which proved useful when it came to implementing the test plan. We had little experience on the testing side of the .NET framework but the implementation of the unit tests ran very smoothly. For mocking and stubbing, the NuGet package manager within Visual Studio provides a package called “Moq”. This package provides everything needed to create mocks and stubs of existing classes and methods.

By the end of the build all the tests were passing and errors were being printed to the console on incorrect input. The test suite executed in a very timely manner, taking milliseconds to complete. The unit tests do not cover all the edge cases of each feature, but they cover enough to inform the developer that the application is sound.

# Difficulties

Requirements analysis was very challenging as we had no contact with the owner to exchange ideas and get a better understanding. We were forced to make some assumptions on the requirements to continue development.

Not being from a testing background, we struggled to create the test plans before the development of the application itself. We found this took a lot of critical thinking to plan features and functions which follow the requirements specification and allow for thorough unit testing.

One aspect of unit testing we struggled with was the use of Mocks and Stubs to keep tests independent of each other. We attempted to include these in the final test suite but failed to implement them before the deadline.

# Reflection

In hindsight, we believe more time should have been spent creating independent unit tests utilizing Mocks and stubs. We spent a lot of time creating the test plans and not enough time developing the actual unit tests.

The simulation itself could have been split into more methods to allow for more coverage of unit tests, this would make it easier to source and fix issues in the application. We would have liked to include testing of getters and setters within the application but time was of the essence.

# Conclusion

Even though mocks and stubs aren’t included in the testing suite, it still tests all the expected inputs and outcomes, along with a range of edge cases.

The application meets the requirements and runs smoothly without error. The tests will show errors during future development, but we highly recommend incorporating mocking and stubbing before development proceeds.