# Wiggle Tech Review

This is an outline as to the thinking that went into delivering this solution.

**Patterns employed**

SOLID

**Application structure**

* Domain project library,
* Domain Models project library,
* Domain project tests
* Project.localization

**3rd party libraries**

FakeItEasy

## Backbone Framework

This is a framework I have been putting together over time. It is still very much a work in progress but has a few nice tools.

I have been working on a few side projects, one being Black hat Development which is the Umbrella name for any side work I do. The framework is intended to facilitate the generic imp mentation of code.

These include: Error handling through notification collections. Extensions methods for strings and objects. Repository patterns.

The Architecture is intended to be loosely coupled and able to adapt to various UI implementations or services. A repository has not been put in place but through e back bone framework one has been made provision for.

I have not used IOC as the scenario tests have been written in a unit test project. Using IOC in a unit test project defeats the whole point of unit tests.as they are meant to be run in isolation. I have not had the time to mock out all the different classes to facilitate IOC. I have instead built the interfaces needed to enable IOC to run in the project, associated them with their relevant concrete classes, but not referenced them as dependencies. I have instead used concreate implementations to enable the scenarios to work end to end.

I chose to make the ProductPurchaseRules class public as this is the access point to the rules strategy governing the discounting of products. However all other rules are internal as they should not be exposed outside the assembly, except through the public methods.

I have not implemented the unit tests for all the classes but rather implemented unit test for one class. Shopping basket dto extension. This choice was down to a time factor.