## [ ] Does the code follow the 10/100 Rule?

* 1. [ ] Added/Changed methods have less than 10 lines
	2. [ ] Changed methods that already over 10 lines are smaller
	3. [ ] Added/Changed classes 100 lines or less?
	4. [ ] Changed class that already over 100 lines are smaller
	5. [ ] Model classes should have zero functions and be closer to 20 lines.

## [ ] Is the code S.O.L.I.D.

* 1. [ ] Single Responsibility Principal
		1. [ ] Does each class have a single responsibility and a specific name?
		2. [ ] Does each method have a single responsibility and a specific name?
		3. [ ] Is this the only class that has this responsibility? (No code duplicated more than twice.)
	2. [ ] Open/Closed Principle
		1. [ ] Can you extend the functionality without modifying the code? Config, Plugins, event registration, etc.
		2. [ ] Is there configuration in the code? If so, extract it. Configuration does not belong in code.
	3. [ ] Liskov substitution principle
		1. [ ] Is “has a” used over “is a”? Inheritance is avoided and only used by architectural design
			1. [ ] If inheritance exists, does the child type avoid causing issues the parent type wouldn’t cause?
	4. [ ] Interface segregation principle
		1. [ ] Does the code use interface-based design?
		2. [ ] Are the interfaces small?
		3. [ ] Are all interface implementations implemented (no empty methods or exceptions thrown)?
	5. [ ] Dependency inversion principle
		1. [ ] Does the code reference only interfaces and abstractions?
		2. [ ] Is constructor injection used?

## [ ] Is the code Unit Tested

* 1. [ ] Is the Code 99% covered?
	2. [ ] Is code not covered marked with the ExcludeFromCodeCoverageAttribute?
	3. [ ] Are tests using proper names: <ClassName>\_<MethodName>\_<State>\_<Result>?
	4. [ ] Are tests written with the AAA Pattern?
	5. [ ] Are all parameter values that could cause different behavior covered?

## [ ] Is everything named correctly

* 1. Are your names typo free?
	2. Do your file names, class names, method names, variable names match existing naming conventions?

## [ ] Big O – is everything performant

* 1. [ ] Are there any glaringly obvious Big O problems? n or n2 vs when it could be constant or log n?

## [ ] Constant handling

* 1. [ ] Are you handling strings safely? No magic strings?
	2. [ ] Are all constants assigned as const variables?

## [ ] Documentation

* 1. [ ] Is every public method/property/field documented?
	2. [ ] Are inline comments avoided in favor of self-documented code?

## [ ] Have you self-reviewed your code?