# Sales Tax Problem Summary

This solution was built as a .NET Core 2.2 MVC app. Simply run the app and use it in a Chrome browser to see the results.

## Design

This app uses a tiered MVC architecture. There is a Controller layer, which relies on a business logic Service Layer, which in turn pulls data from a Repository layer. All of these layers are implemented through interfaces and dependency injection to allow for more decoupled layers. The app also makes use of Type parameters at the service layer to allow for multiple implementations of a single service to be registered with the dependency resolver.

The user interface is a simple web page. This page is intentionally minimalistic to avoid the use of powerful frameworks like Angular or React, or even jQuery. This would be the biggest area of potential improvement. There is some model validation at work on the front end if you enter in bad data, but for the most part the UI is pretty static to allow you to enter in the sample data.

The data layer is simulated using singleton registration with dependency resolver, so it acts as an in-memory data store. A real app should obviously use some type of database and potentially an ORM like Entity framework. Due to the loosely coupled interface design, you could easily swap out that layer without effecting the rest.

## Running the app

To run this app you’ll need the latest versions of the **.NET Core 2.2 SDK and Runtime**, and the latest version of Visual Studio. .NET Core updates rapidly, so it’s easiest to just build new projects off of the latest versions to avoid many potential conflicts.

## Code Quality

A few measures have been taken to help with code quality:

**StyleCop**: The app includes stylecop enforcement for syntax, code comments, and general structure. There should be no Stylecop errors on compilation. Code comments have been used for class member documentation, and used sparingly inline for helpful info in various spots.

**Unit Tests:**  Unit tests for some of the components are provided to give an idea of what the tests might look like for Controllers, Service, Helpers, etc. The tests are not meant to be a comprehensive suite for maximum branch coverage, but rather a proof of concept to demonstrate writing unit tests.

## Note on 3rd Party Libraries

All of the logic in the app is written using System.\* or Microsoft.\* DLLs and the out of the box MVC Visual Studio template. However, I did utilize Bootstrap for some light styling on the UI to make it more visually appealing, since it’s included in this base template. I figured this would be acceptable since there is no logic driven by bootstrap.

I also did include Moq and NUnit for the Unit Tests. Since this is not part of the core application requirements I figured this would also be acceptable rather than writing a custom Mocking library for a few unit tests.

## Demo Video

A short demo video has been included in the zip file. This simply shows a sample workflow for entering all the scenarios, and illustrating the desired outcomes.