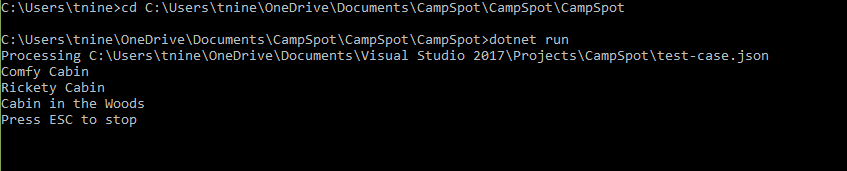
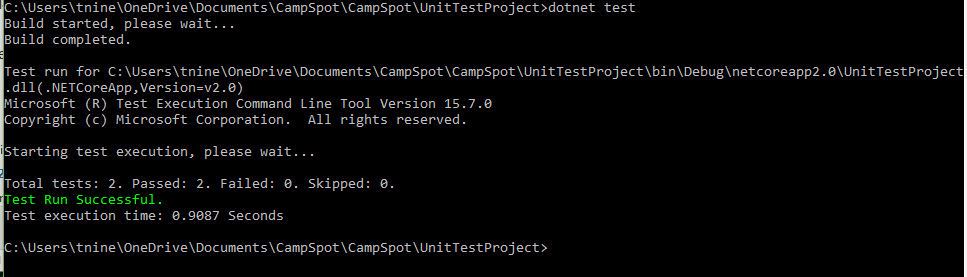
# Documentation

To run the Campspot program extract the zip file and navigate to the campspot project of the campspot solution and type *dotnet run*. This runs the source code of the current directory against .net core on any operating system where .net core is installed. See example below.



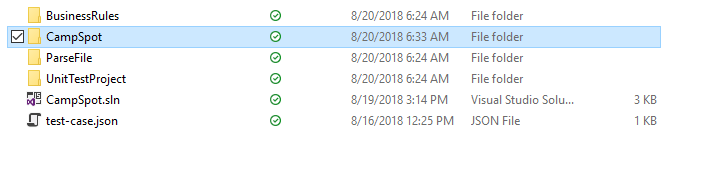
To run the Campspot tests navigate to the unittestproject and type *dontet test*. See example below.



# Design Discussion

My experience with .net core has been limited to a project for migration from legacy code (web froms) to an asp.net mvc core web application. So, I thought it might be fun to do a console application in asp.net core. Note console applications are seldom used in the wild except for prototyping or internal toolsets.

## Program Organization



The CampSpot solution contains 4 projects. BusinessRules is meant to contain business rules and is where the GapRule is implemented. Campspot is a simple console application which uses BuinessRules and ParseJson project to create a display of allowed reservations based on the input json file. ParseJsonreads has methods to read json file and to create a reservation data structure for searching. UnitTestProject has the start for unit tests for the project.

## Deserialization of JSON to classes

### ReadJsonFile.cs

This approach was taken due to:

1) simplicity

2) ability to add extra functionality to classes and marking with attributes for serializer to ignore [JsonIgnore], in this case a day of year property.

3) ability to leverage Linq for navigation. Linq is Microsoft’s Lord of the Rings language concept one language to rule them all – a common query language regardless of datasource.

4) easy generation of classes from your favorite site such as <http://json2csharp.com/>

5) easy to make generic reader

## Dictionary data structure with list of reserved days

This approach was taken for simplicity in searching for gaps with an entry for each day. For simplicity this is limited to current year. It is assumed that the persistent mechanism for storage is driving this design in real life.

### LoadReservation.cs

Given a list of campsite ids and the root class containing search and collections of campsites and reservations return a dictionary indexed by campsite and a list of reservations for each campsite. A list of campsites is passed to LoadReservationsList as it seems this may be limited by metadata upstream by serach criteria such as show me only campsites where I can park my monster 200 foot motorhome, etc..

## GapRule

Given a dictionary a dictionary indexed by campsite and a list of reservations for each campsite, a list of campsiteids, the start date and end date, and a gap find a valid reservation.

### GapRule.cs

Check first to see if we can reserve the day and then check both sides of the date to see if a gap exists.