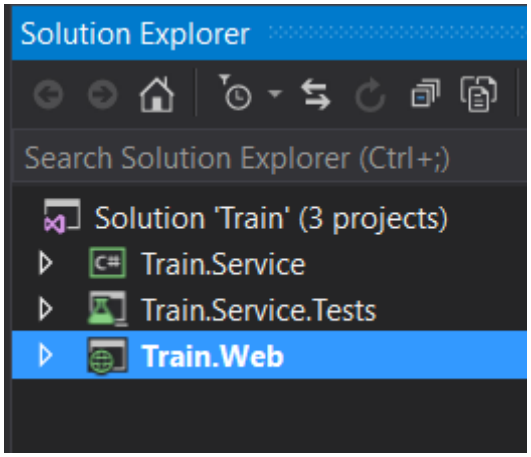
Architecture:

A simple MVC application (1 page only) to call a RESTful API (ajax call).

The route finding algorithm is customized Breadth-first search. I had to modify this classic algorithm to answer all 10 queries. This algorithm sits in Train.Service.csproj

Solution Explorer in VS2015:



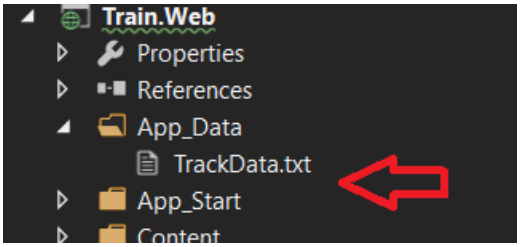
I used ***Unity*** as DI container.

Data:

For the purpose of unit tests, graph algorithm takes the tracks manually.

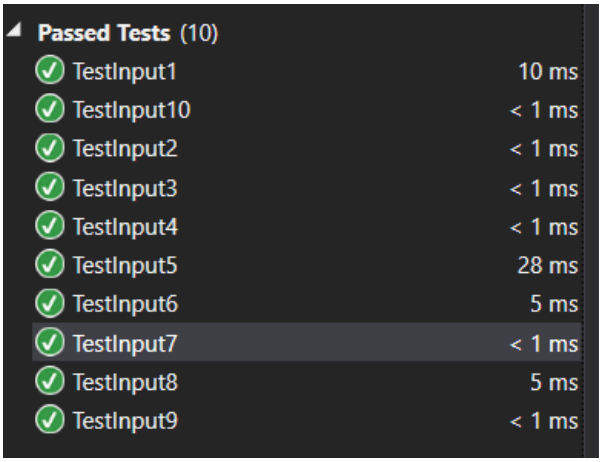
```
[TestClass]
public class TownGraphTests
{
    private ITrainQueue _queue;
    private ITrainGraph _graph;

    [TestInitialize]
    public void Setup()
    {
        _queue = new TrainQueue();
        _graph = new TrainGraph(_queue);

        _graph.AddRoute(Track.NewTrack("AB5"));
        _graph.AddRoute(Track.NewTrack("BC4"));
        _graph.AddRoute(Track.NewTrack("CD8"));
        _graph.AddRoute(Track.NewTrack("DC8"));
        _graph.AddRoute(Track.NewTrack("DE6"));
        _graph.AddRoute(Track.NewTrack("AD5"));
        _graph.AddRoute(Track.NewTrack("CE2"));
        _graph.AddRoute(Track.NewTrack("EB3"));
        _graph.AddRoute(Track.NewTrack("AE7"));
    }
```

However, UI layer read tracks from a text file (as requested) and store it in Cache. Track data file exists in App_Data folder of UI project.



Tests:

10 MsTests added in Train.Service.Tests.csproj which checks answers out. I used *FluentAssertions* for assertion.



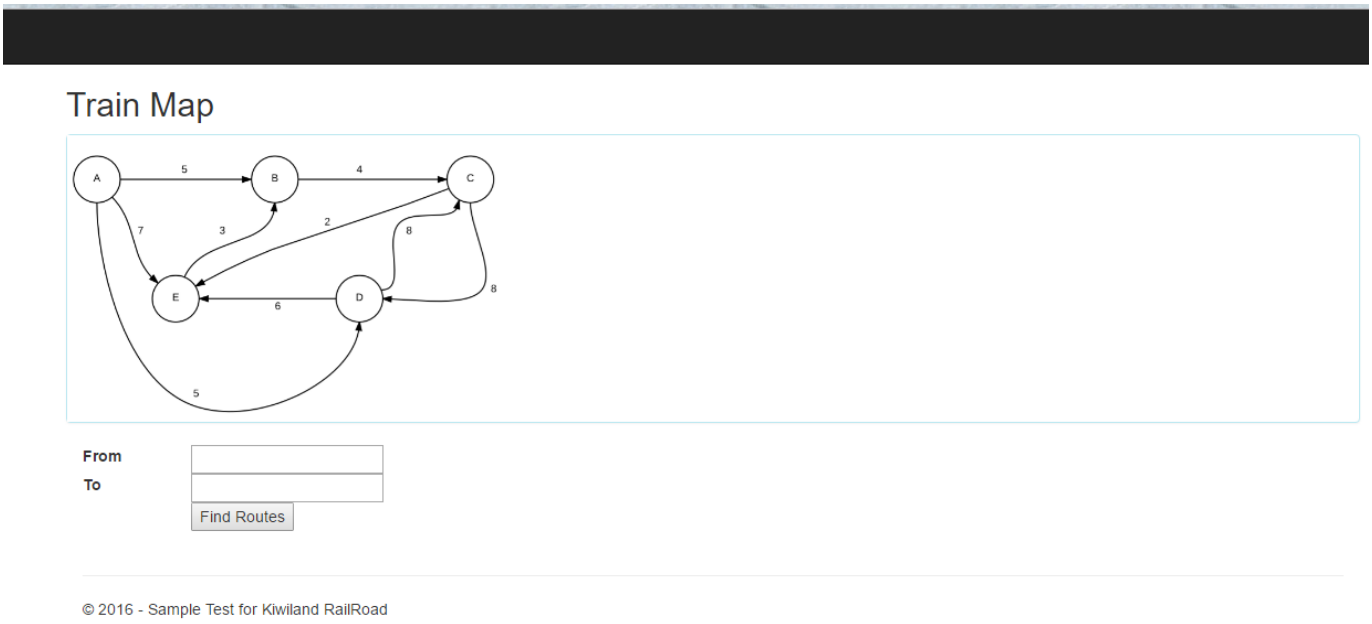
UI

Assumptions:

1. Although the expectation for this test is production quality code, I found that I have to spent days to make it ready by having unit test coverage of over 70%, UI automated test and also more separation of code and/or better UI. Thus, I implemented basic UI and essential tests to make sure we have a prototype and not production quality product.

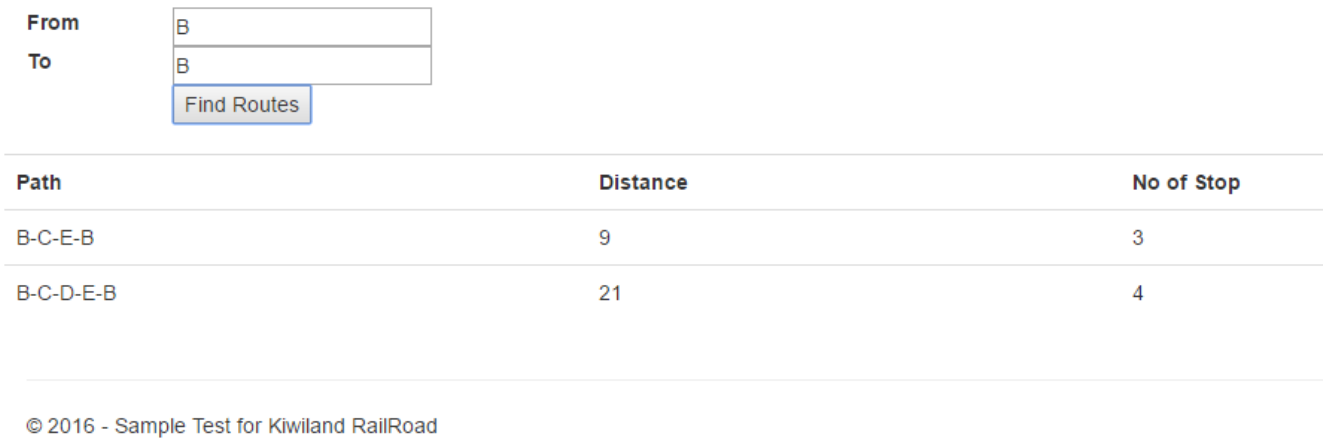
2. This UI is only tested on Chrome.

After launch of Train.Web, home page will look like:



User can enter the name of starting/ending journey and all route (with no circular trip as mentioned in question 10) will be displayed. The assumption here is that a circular journey for a commuter is not realistic.

An example of UI for question 9 would be:



Steps:

1- Unzip the solution into your hard drive.
2- Open Train.sln
3- Build the solution
4- Make sure all unit tests run and green.
5- Run Train.Web