DotNetCodeKataSample:

1. **Test Description**

Use .NET for solution. The goal of this exercise is to get a better glimpse into your thought process. While this is a simple exercise, think of it as a large project, so put whatever patterns you think would be necessary for an enterprise level application. Create a ReadMe file to explain any details you think would help verify your work. Submit solution to GitHub or any other publicly accessible code repository. The code will process an input file. Each line in the input file will start with a command. There are two possible commands. The first command is Driver, which will register a new Driver in the app. Example: Driver Dan The second command is Trip, which will record a trip attributed to a driver. The line will be space delimited with the following fields: the command (Trip), driver name, start time, stop time, miles driven. Times will be given in the format of hours:minutes. We'll use a 24-hour clock and will assume that drivers never drive past midnight (the start time will always be before the end time). Example: Trip Dan 07:15 07:45 17.3 Discard any trips that average a speed of less than 5 mph or greater than 100 mph. Generate a report containing each driver with total miles driven and average speed. Sort the output by most miles driven to least. Round miles and miles per hour to the nearest integer.

Example input:

Driver Dan

Driver Alex

Driver Bob

Trip Dan 07:15 07:45 17.3 Trip

Dan 06:12 06:32 21.8 Trip

Alex 12:01 13:16 42.0

Expected output:

Alex: 42 miles @ 34 mph

Dan: 39 miles @ 47 mph

Bob: 0 miles

1. **Solution Details**

This .Net solution consists of angular web application & .Net Class library implemented in Singleton & FactoryMethod design patterns.

* **Angular Web Application**

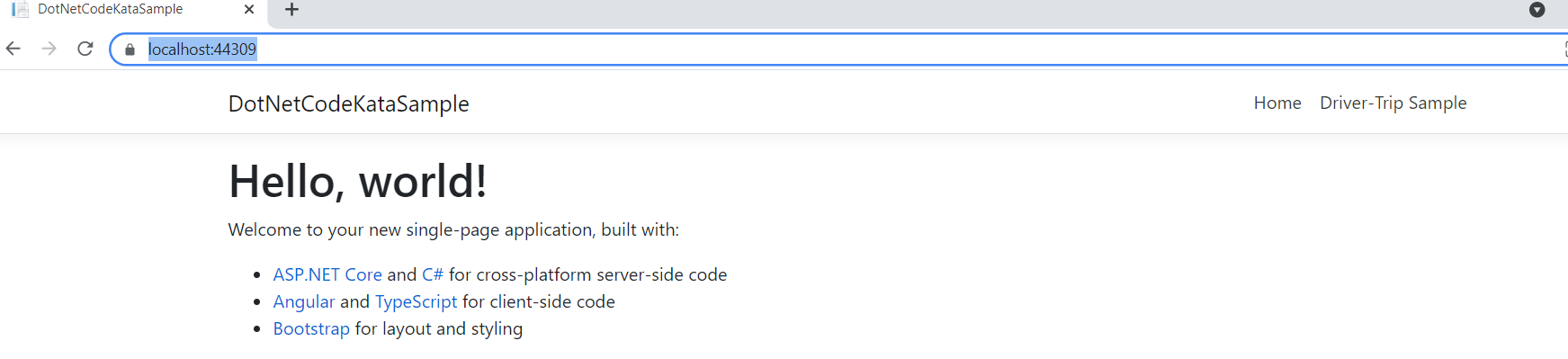
This web application is created by using .Net Core in VisualStudio.

.Net Core 3.1

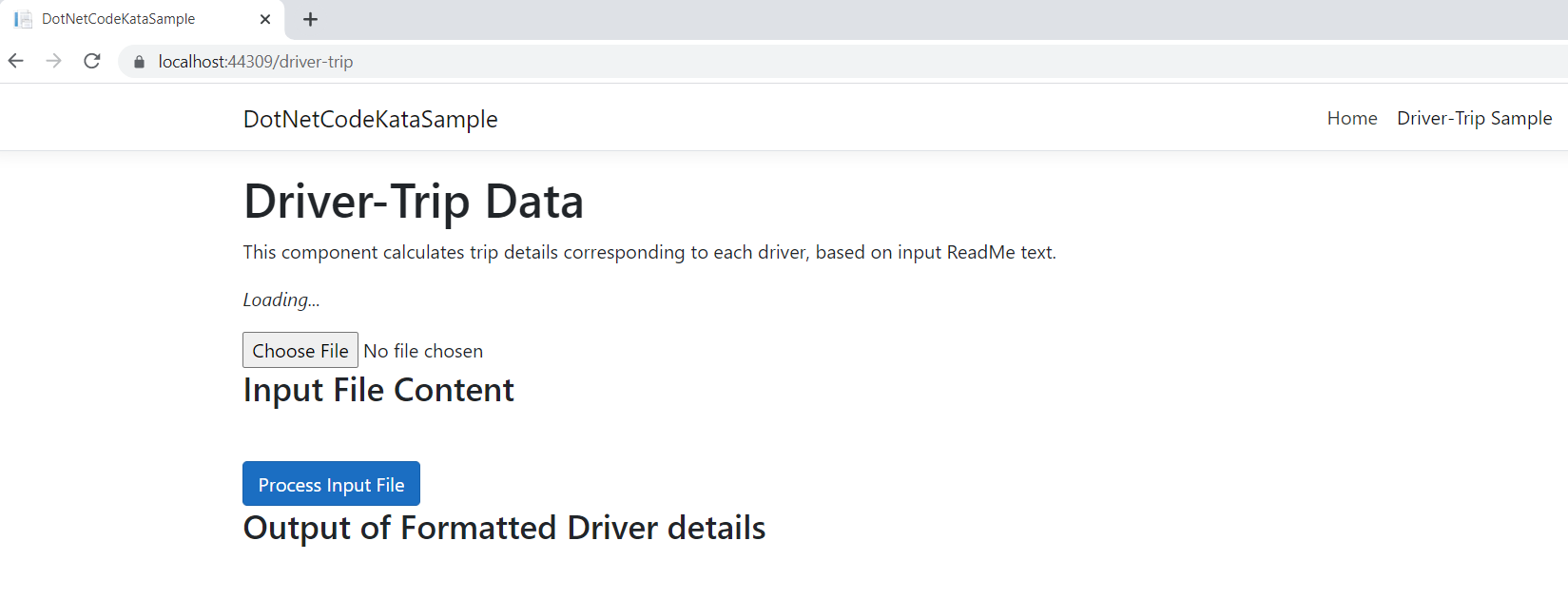
VisualStudio 2019

ES 5 for TypeScript

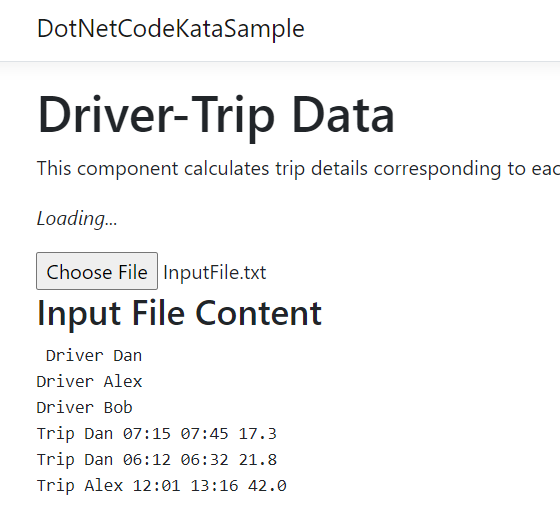
On running this web application, it loads with Home page and link on right corner **Driver-Trip Sample** to load the component of solution provided for CodeKata.



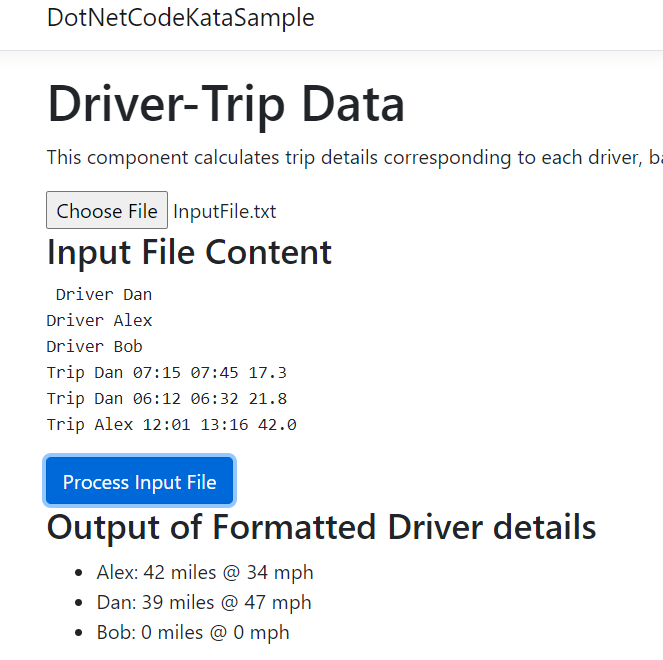
A new component has been included D**riverTripComponent** with TypeScript coding and corresponding HTML file; to provide solution for CodeKata and has been imported in **app.module.ts** file.



On clicking Choose File input button, FileDialog opens to select valid Input file (attached in email) and display the file contents



On clicking Process Input file button, corresponding processed driver-trip details are displayed



The TypeScript code will access the .Net Class Library using POST command supported by Controller & BackEnd code (Class Library)

* **.Net Class Library**

This class library has been implemented using both Singleton & FactMethod design patterns. It includes separate classes for Driver, Trip & Driver Details (shared to frontend). Both Driver & Trip details are processed using Factory Class using Dependency Inversion principle guidelines.

1. **Shared link**

I couldn’t upload the entire project into GitHub (free account), due to storage limit issues. Please find below GoogleDrive link to download the entire .Net solution.

[**https://drive.google.com/file/d/1otBHCAZngc9zU6sgNmqJRlxaY8Gp9ijf/view?usp=sharing**](https://drive.google.com/file/d/1otBHCAZngc9zU6sgNmqJRlxaY8Gp9ijf/view?usp=sharing)