Technical Interview Take-home Test

Step 1

Write a program that processes a JSON file located in the current directory named ‘data.json’ with the following structure (this is just an example, and a different file using the same structure will be used to verify the submitted program):

```
{
  "datasets": [
    [1.4, 2, 3, 4],
    [2, 3.7, 4],
    [7, 8.87, 9, 10.01, 11]
  ],
  "generators": [
    {
      "name": "Gen1",
      "interval": 1,
      "operation": "sum"
    },
    {
      "name": "Gen2",
      "interval": 2,
      "operation": "min"
    }
  ]
}
```

The objects in the *generators* array define an operation to perform: sum, average, min, or max. Each generator must perform that operation on each array of values defined in *datasets* and write that value to stdout including the current timestamp (which must show seconds resolution), the name of the generator and the result of the operation. Generators must wait for *interval* seconds between writing each result. All generators must start at the same time.

As an example, the above dataset would result in the following output:

```
13:53:39 Gen1 10.4
13:53:39 Gen2 1.4
13:53:40 Gen1 9.7
13:53:41 Gen2 2
13:53:41 Gen1 45.88
13:53:43 Gen2 7
```

The program is to be written in C# (any version of .NET) and the complete source including solution and project file must be packaged into a zip file for submissions. Do not include any binary files in the zip.

Step 2

Create a WPF application that allows the user to load a JSON file (with the same structure as in Step 1 above) and view both the datasets and generators. The view must use WPF controls and not simply display the JSON text. The UI should implement responsive design that responds to changing window sizes.

The user should also be able to fully edit the datasets (both the values and number of values) as well as the interval and operation of the generators. It is NOT required to be able to add or remove either datasets or generators.

The user must also be able to run the generators and see the results of the run as in Step 1 in a textbox in realtime (NOT in a console window).

The application should be written using the MVVM pattern with a viewmodel class to contain all UI logic and little to no code-behind. It should also reuse the code created in Step 1.