Write an ASP.NET Core, single-page application (using Angular as a front-end) with Web API back-ends to manage the processing of numbers using Visual Studio (free version available online). The requirements for this system are below. Although this is a contrived example, this is your chance to show your understanding of design patterns and practices.

**Web App**

* The application will require the user to enter two numbers between 1 and 10
  + The first number, X, is how many batches of numbers should be processed
  + The second, Y, is how many numbers will be processed per batch
* A start button will be available to click once the input is ready
* Once started, the application will trigger back-end work
  + This is to be done in parallel, and not sequentially
* A grid of the batches, their remaining numbers to process, and their current totals should be displayed
  + The grid should update every 2 seconds.
  + Use polling through Angular and not through a third-party library like SignalR
* A grand total (sum of all batch totals) should be displayed
* Once all batches are processed, the user can start another batch, clearing previous results
* Persist these results, using Entity Framework (and the In-Memory provider)
* Allow the user to retrieve and view the **last** batch on a separate page in a grid
* There is no need to use queuing such as MSMQ or Rabbit etc.

**Web API**

* An endpoint will exist to start processing X batches with Y numbers per batch managed by a Processor
* The Processor will contain two workers to manage the processing by listening to events raised from its workers: GeneratorManager, MultiplierManager
* The GeneratorManager will request generated numbers from a separate service
* The MultiplierManager will request multiplication from that same service
* For each batch, the Processor will ask the GeneratorManager to request X new numbers
  + When the GeneratorManager receives a number, **it will raise an event for the Processor identifying the batch and number**
* For each generated number in a batch, the Processor will ask the MultiplierManager to multiply the number
  + When the MultiplierManager receives a multiplied number, it will raise an event for the Processor identifying the batch and number.
* The Processor will take each multiplied number for a batch and aggregate them as received
* An endpoint will exist for retrieving the current processing state
* An endpoint will exist to relay a generated number for a given batch to the GeneratorManager
* An endpoint will exist to relay a multiplied number for a given batch to the MultiplierManager

**Secondary Web API**

* An endpoint will exist to start number generation for a given batch managed by a Generator
* The Generator will generate Y random integers between 1 and 100
  + For each number, a random delay of 5 to 10 seconds should be used to simulate work
  + The generated number should be returned to the web application via its endpoint
* An endpoint will exist to start multiplying a number for a given batch managed by a Multiplier
* The Multiplier will multiply a number by 2, 3, or 4 (chosen at random)
  + A random delay of 5 to 10 seconds should be used to simulate work
  + The generated number should be returned to the web application via its endpoint

**Some notes**

* Yes, we get it, this is a contrived example, but it is your chance to show your understanding of design patterns and practices.
* Please provide your solution in a single, archived file
* It should contain at **least** two, separate projects for the APIs mentioned below as well as any other organisational projects or unit test projects you use
* Only files needed to build and run the application should be included (i.e., **no build artifacts**).
* Any assumptions or decisions made that should be considered during review can be included in a README file or as comments inline as appropriate
* If there are any questions about the requirements, reach out as needed
* Use as much of the .NET Core base libraries as possible and minimise dependencies on external libraries
* Make it as easy as possible for us to run your code, ideally Start > Run from within Visual Studio (i.e., no Docker, Rabbit MQ or similar prerequisites)