Current Implementation

* The program depends on 3 main items
  + Anagram Tool: to get the anagrams from the file
  + Numbers Tool: to get the numbers from files and get the factorial task
  + File Handler
    - File Read
    - File Write
* The implementation depends on abstraction; which means all the 3 main items –listed above- are injected using Dependency Injection
* For that, If we need to read from another source –e.g.PDF- we can implement the IFileRead, and IFileWrite and inject them in the code without any changes in the current code

How would you change the design if the input file was very large?

The current implementation depends on abstraction, so if we need to implement another algorithm to get anagram, we can do it and inject that.

If the file was very large, as we here depend on the StreamReader and we read the file Line by line and not load it in the memory one time, so I guess the current implementation will work.

And we can depend on RedisCache in the cloud to save the anagrams instead of the in memory Dictionary.

How would you change the design if the input data was bigger than the available RAM?

**Cloud Solution**

And we can rely on the Cloud services if we need, we can use Azure Functions and add the file in a storage and read it and save the result in another storage file

We can add the file in Azure storage and run the Azure Functions or background worker to fetch the data and process it.

**Data Tools**

We can use BI tools or Big Data tools to deal with the big files