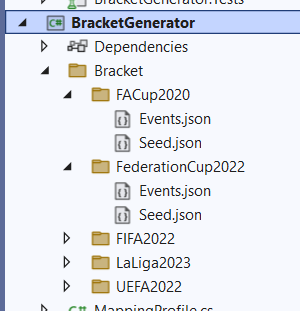
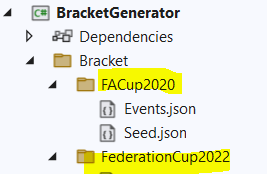
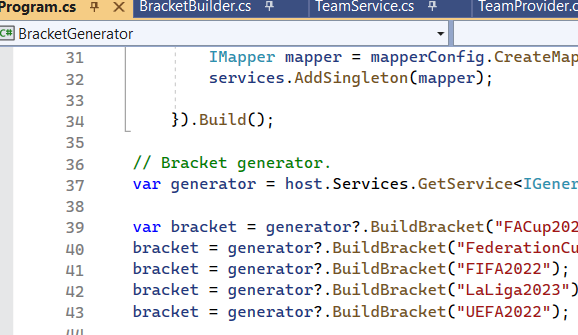
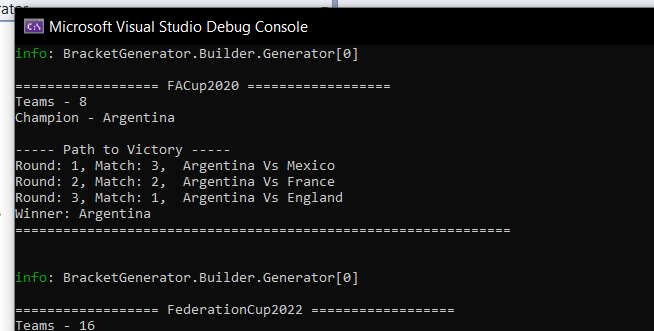
# Implementation details.

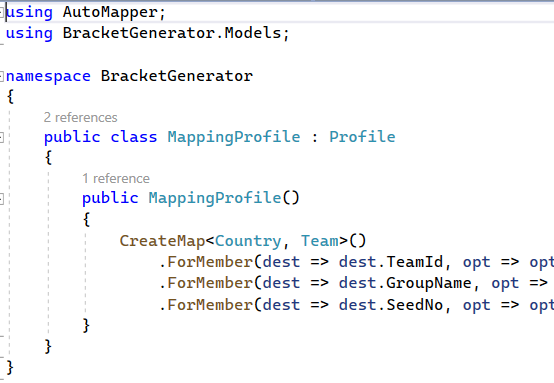
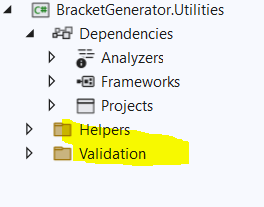
* This implementation supports, for both group and group less tournaments.
* Also, for now it is only possible to work with the numbers of power of 2.
* Implementation does not support for biases.
* As an example, 64 teams support groups of 32, 16, 8, 4, 2.
* Implementation support any number of teams and groups under above mentioned conditions.
* I have implemented my own formatted seed and JSON files.  
    
  
* When you add seed and event file properly, it is possible to generate everything by tournament name that means the folder name. Refer the “Program.cs” file to build the bracket.  
    
   
* Implementation is fully executable, and output can be visible in the console.  
    
  
* From the code it is possible to fetch all the data related to teams, how many rounds and each team have played, round of elimination, opponent details…etc.…
* Considering unit testing I could not be able to cover all the scenarios due to time limitation as I have dropped the weight on implantation. But of course, the written tests are successfully passed.  
    
  A screenshot of a computer

  Description automatically generated

# Technical Details

* Project was implemented with a layered architecture breaking down the subsystems which is compatible with “Façade” design pattern.  
    
  A screenshot of a computer

  Description automatically generated with medium confidence
* Dependency injection was used all over the project.
* Building a bracket was implemented according to “Builder” design pattern.  
    
  A screenshot of a computer

  Description automatically generated with medium confidence
* SOLID principles were applied with in the implementation. Interfaces and abstract classes were used. It is possible to use virtual methods, partial classes if we need further enhancement which depends.
* “Auto mapper profile was added to the project easily map the entities.  
    
  
* Couple of helper methods were implemented to support the project that can be easily used when needed.
* Validations were added using a separate section that can be injected through DI and use anywhere.  
    
  
* It is essential to add the code comments and breaking regions, but due to time limitation I could not add them.
* Newton JSON library was used to deserialize the data from files.
* Considering unit testing “Fake it easy” library was used to mock the objects.
* Please rebuild and execute.