**Football Exercise**

* The solution finds the team/teams with the minimum difference in for and against goals

The solution contains 5 projects

|  |  |
| --- | --- |
| Project Name | Description |
| EnglishPremierLeague.Console | Console application used to run the application |
| EnglishPremierLeague.BusinessServices | This project is the business service layer to obtain the team with the minimum difference between for and against goals |
| EnglishPremierLeague.Data | This project is the data repository layer to obtain the data from the csv/dat |
| EnglishPremierLeague.Common | This project is used to have the common entities used across the other projects |
| EnglishPremierLeague.Tests | This project contains the integrations tests. |

**Design considerations:**

* Should cater for csv and dat file format
* Columns order and column names should be modifiable without change in code (create xml templates for each csv and dat file formats)
* Should follow SOLID principles
* Should be testable
* Should be robust
* Modular
* Should be modifiable for different business requirements
* Should be able to accommodate any new file formats with minimal change in code.
* Should also be able use the logic if data is obtained from database.

**Technical toolset used:**

* .NET CORE Console application
* Dependency injection in .NET CORE framework
* Logging framework in .NET CORE framework
* Xunit
* Moq ( Time constrained and was not able to write mock tests)

**Patterns Used:**

* Adapter pattern
* Template pattern

**Data Validations considered:**

* Column name validation
* Column width validation
* Row data type validations
* Invalid row validations

**Business Validations**

* Total played = Wins + Losses + Draws
* Points = (Win \* 3)+ (Draws \*1) ( The given file did not have the data right)

The project can be run by traversing to the dll path in command window or powershell.

**Improvements required.**

* More unit tests – (Covered the basic integration tests but did not cover the detailed unit tests and functionality tests using Moq)
* Move some of the logic to base class to prevent redundant code in child classes.

USAGE:

---------------------------------------------------------------------------------------------

dotnet EnglishPremierLeague.dll [-csv|-dat] -filepath <"filename.csv"| "filename.dat">

[-loglevel [None|debug]] [-csvtemplatepath|-dattemplatepath <"csvtemplate.xml">

---------------------------------------------------------------------------------------------

-csv : Use CSV functionality for the input file

-dat : Use DAT functionality for the input file

-filepath : Full path of the input file within double quotes.

This option is mandatory.

-loglevel : Loging level for the program [debug | none]

-csvtemplatepath : Full path of the csv template within double quotes

-dattemplatepath : Full path of the dat template within double quotes

---------------------------------------------------------------------------------------------

default values:

-csv

-loglevel : none

-csvtemplatepath : Template within the project

-dattemplatepath : Template within the project

---------------------------------------------------------------------------------------------

Examples:

dotnet EnglishPremierLeague.dll -csv -filepath "C:\Temp\Data.csv"

dotnet EnglishPremierLeague.dll -csv -filepath "C:\Temp\Data.csv"

-loglevel debug -csvtemplatepath "C:\Temp\CSVTemplate.xml"