**Australian Open Analytics Solution**

This document briefly outlines the project structure and technology used, in the first section. The second section explain the solution to some details.

# Project Structure & Technology

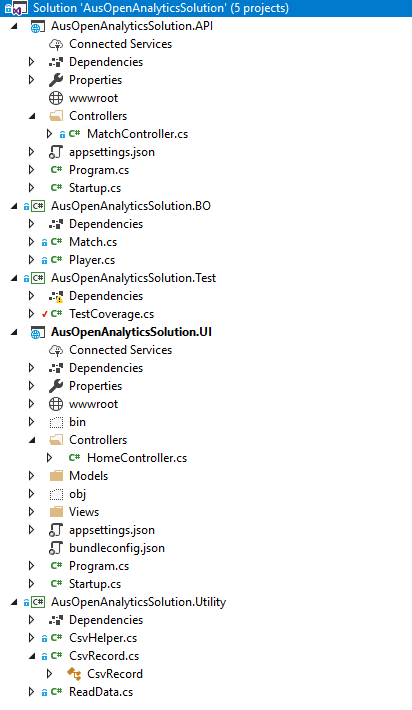
**Please note: The current solution is only viewable through browser supporting SSE (Server Sent Event), including Google Chrome, Firefox, and Safari. However, it will be not supported in IE/Edge.**

GitHub location: <https://github.com/aschattopadhyay/AusOpenAnalyticsSolution>

The solution is built using Visual Studio Enterprise 2017 Version 15.5.5. It uses the following technology stack:

* .Net Core 2.0 server-side infrastructure (i.e. Web API and MVC) and
* jQuery client side using D3.js for plotting

In Visual Studio it looks like below:



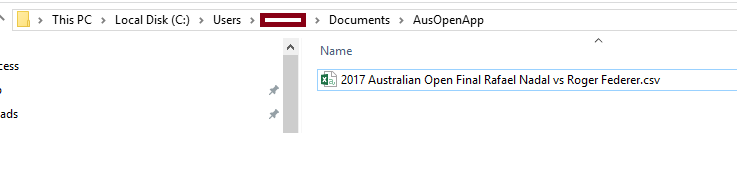
* **API** project the Web API projects built on .Net Core 2.0. This API streams data
* **BO** is the Business Object projects defining 2 business objects – Match and People
* **Test** is the Unit Test project
* **UI** is the MVC project built on .Net Core 2.0 and uses jQuery/D3 as front end
* **Utility** provides some utility methods e.g. reading data, defining a class titled CsvRecord.
* The data source for the projects Is a CSV File titled *2017 Australian Open Final Rafael Nadal vs Roger Federer.csv*, which is stored inside My Documents folder of the logged in user.

The solution focuses on one match only which is the 2017 final between Federer and Nadal. The idea is the same programming logic can be applied to other matches too.

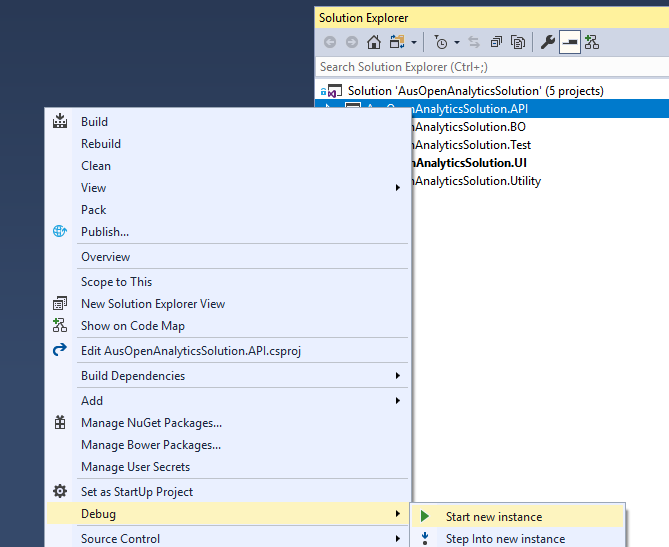
As this is a Prototype, few assumptions have been made which will be highlighted in the solution section.

# Solution

To get started, a CSV data source for the relevant match (2017 Aus Open Final between Roger Federer and Rafael Nadal) has been prepared. The data source has been titled *2017 Australian Open Final Rafael Nadal vs Roger Federer.csv*. **Make sure you put the file under C:\Users\<Logged in user>\Documents\AusOpenApp**, Logged in user being the person who is running the application.



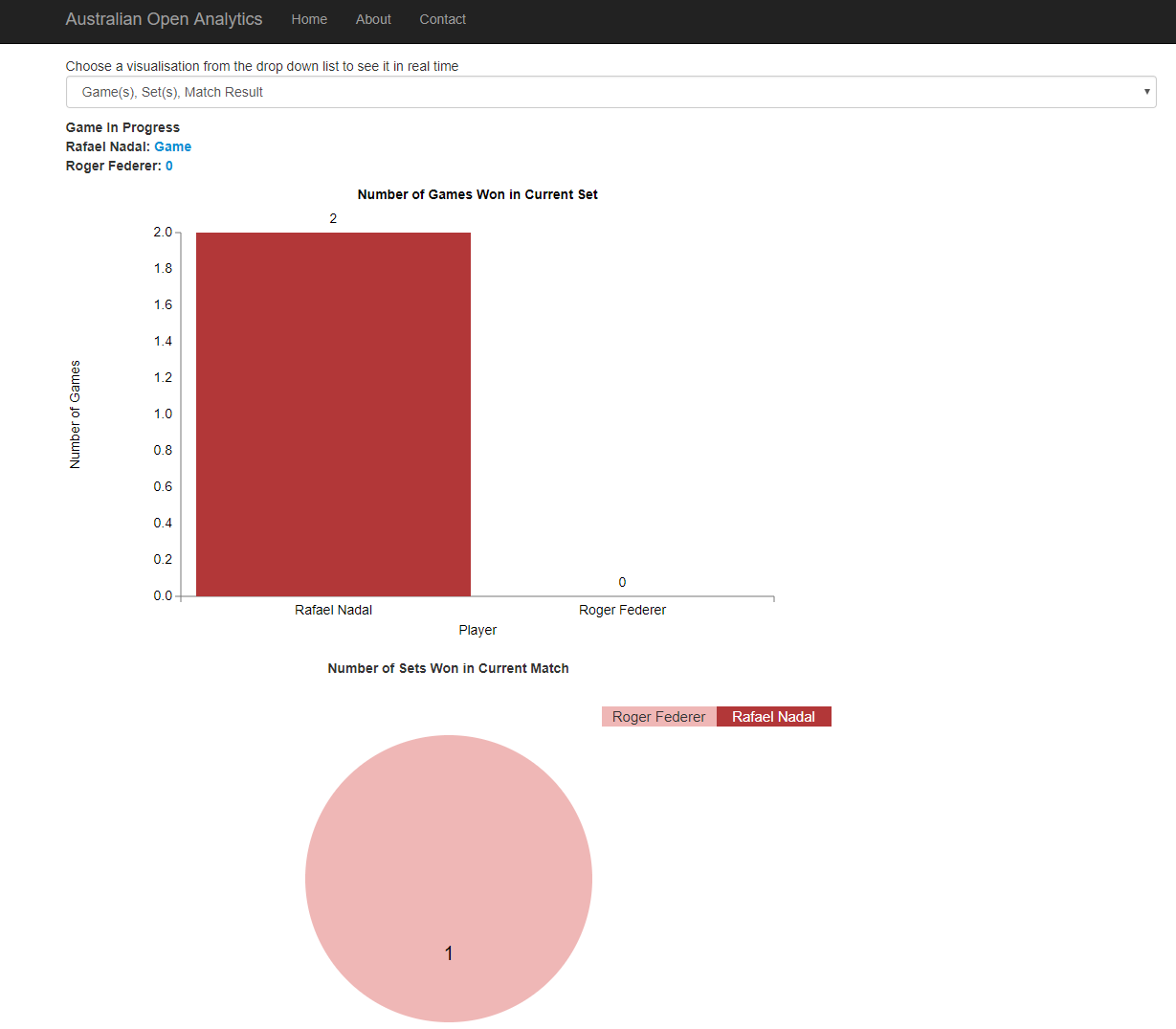
After this get the solution for the GitHub repository and open in specified version of Visual Studio 2017 (as Administrator). Build the solution. Right click on API project -> click on Debug > click on Start a new instance. After that right click on the UI project -> click on Debug > click on Start a new instance. The API always runs on port 3703.

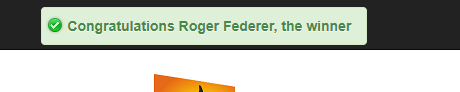


**Assumptions & Explanations**

* For the purpose of project, the CSV file titled *charting-m-points.csv* has been considered as it gives the match data, point by point. From that file, a smaller file relevant to this project’s match (2017 Aus Open Final between Federer and Nadal) was prepared. The file is titled *2017 Australian Open Final Rafael Nadal vs Roger Federer.csv.*
* For every visualisation type, the assumption is the match starts from point 1 and goes ahead till the last point when the winner is announced. So, every time a new visualisation is selected from the dropdown, the streaming starts right from point one.
* Five types of visualisation have been provided:
  + **Game(s), Set(s), Match Result**

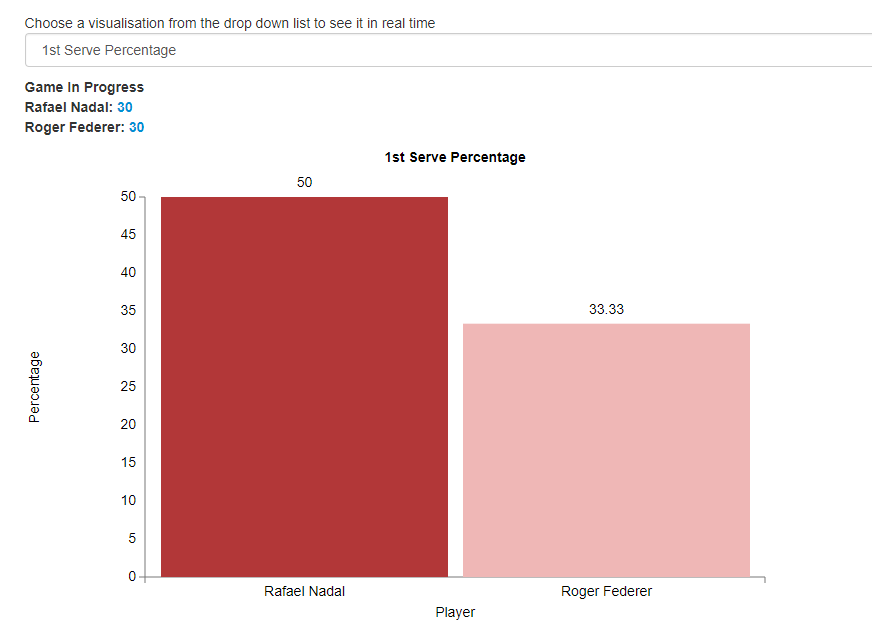
It shows the points won by each player for the game in progress (the point updates on the screen), number of games won in the current set, and number of total sets won by each player. In future it can be further extended to show number of games won in every set by each player.



At the end of the match, the winner is announced with a green notification at the top.  


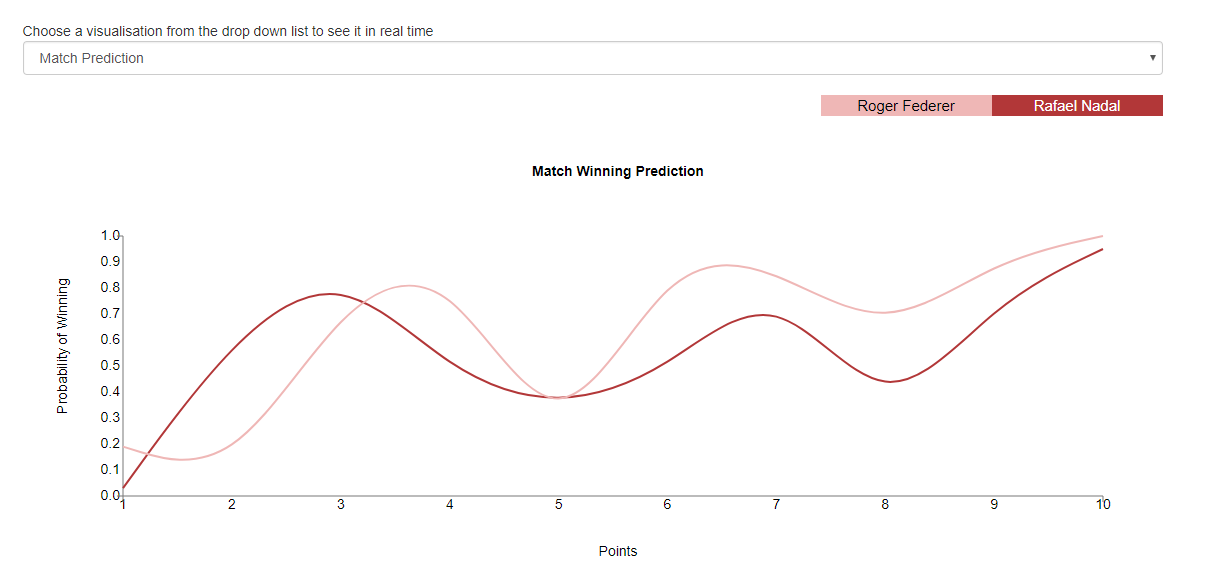
* **1st Serve Percentage**

This one shows the % of 1st serve for each player, as the game progresses. 1st Serve Percentage means percentage of 1st serves which are considered to be in.



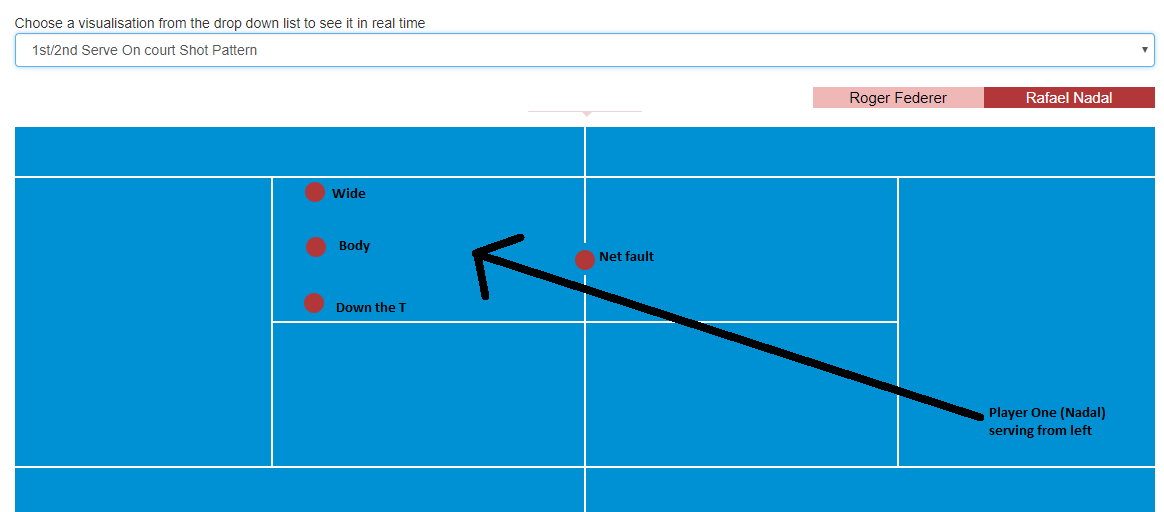
* **Match (Winning) Prediction**

This one plot the winning probability for each player, for every point. Currently a random probability (between 0 and 1) is generated for each player. In Ideal something like what is mentioned in this [research paper](https://pdfs.semanticscholar.org/114a/2c60da136f80c304f4ed93fa7c796cc76f28.pdf), should be implemented to calculate winning probability. At the very end, the probability of the winner becomes 1 while for opponent It becomes 0.



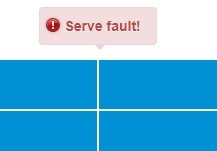
* **1st/2nd Server On Court Shot Pattern**

This one shows the on court short pattern of 1st and 2nd server (if any) for every point. The shot patterns are assumed to be shown like below:



The image above shows court pattern for Player One (Nadal) serving from left. Similar assumptions can be made for each player, depending in the direction (left or right) of the serve.

In terms of fault, only net fault is shown. Everything else is shown as serve fault, with a notification like below.



This visualisation makes use of Simple Vector Graphics (SVG). The plot also shows probability of wide on 2nd serve from left, for each player.

* **Number of Aces**

This one shows the number of aces for each player, as the game progresses.



**Future Enhancements**

* In ideal world, the CSV data should reside inside a database server.
* To predict match winning probability, something like what is mentioned in this [research paper](https://pdfs.semanticscholar.org/114a/2c60da136f80c304f4ed93fa7c796cc76f28.pdf), should be implemented.
* The solution can also be integrated with <http://accord-framework.net/>, which is a .NET machine learning framework combined with audio and image processing libraries completely written in C#. It can be used to predict things like below:
  + - Whether the player will do a second server in the current point
    - Whether the player will do a net fault in the serve

etc.

* Granular Exception handling

As the task was a prototype, mathematical implementations need to be re-visited during the time of productionising it.