Test Report

Overall there was quite an extensive set of tests conducted throughout the System to verify all functionality were running flawlessly, error free and bug free. The Testing was conducted via the following use-cases:

* Unit Testing
* Integration Testing
* System Testing

The majority of functionality was integrated on the Server side, more specifically within the API Controllers and the corresponding models. These were the core part of functionality and a lot of validation and algorithmic functions are found here. To test the API controllers and models a series of Unit Tests were performed to ensure all the methods worked correctly I conjunction with their relative models.

In total over 40 Unit tests were carried out across all the API controllers. The following tests were conducted:

* GET – Return Team with Same ID (comparison)
* GET – Return all Teams checking for Nulls and successful comparison
* POST – Create Team and return with comparison
* PUT – Update a Team with a specific ID (e.g. /5) ensuring no status codes are thrown and a comparison of the change is successful
* PUT Failure – Performing an update but with different ID’s. Comparison will be different and the Test should pass should a Bad Request be picked up.

The Unit Tests above covered the main API methods that were implemented throughout the project. It was important all of these were tested to ensure they were all functioning correctly. If no testing was done for these sections there may have been potential bugs within any element of the API and errors may have arisen that could have potentially broken the whole system or caused an incorrect calculation of the UEFA Ranking System.

The testing was quite extensive and a lot of time was allocated to generating them and covering all possible approaches that can be made to the API that could break it. The Unit testing was quite complex due to the fact the API works with an Entity framework. In order to perform Unit Testing on the API the Entire Entity Framework had to be mocked. All the controllers had to be modified to enable the passing of a context object for testing. Custom Test Controllers and a mock interface of the Database Context had to be created in order for the Unit Testing to be conducted. This was quite a complex piece of integration however it was essential the API was fully tested as it is the Core functionality to the whole System.

Testing was also conducted for the MVC end. All of the Model Classes contained validation and some regressive testing was conducted from the Administrator Web Client to attempt to break the Database through inserts. Several inputs were tried and were all picked up by the validation.

Tests were conducted against the SSL (HTTPS) ensuring the MVC Web Service could not be accessed through a normal HTTP connection. There were also tests done against a normal user and guest attempting to access the Admin only sections. The Admin has rights to all PUT, POST and DELETE methods within the API and the Web Service. Any sort of access from a normal user and guest was blocked from a series of tests. Currently all attempted access will redirect the guest/user to the login field.

System testing was conducted for the front end of the client. Several attempts were made in various ways to try breaking the Application.

The attempts to break the application included the following:

1. Disconnecting the Internet connection – Before opening the app, during an async call, when on a page containing Server Side data.
2. Multiple taps on Intents – Multiple selects on a single button or various buttons at once were attempted and no problems occurred.
3. Returning and Re-entering intent – Accessing the 1st round API and the previous menu over and over were done. Each time this was attempted the errors were caught and the user was returned to the previous screen.

The App has been regressively tested and no errors have been found anywhere. The app has been made available to 5 clients who have all tested the app without any errors occurring during their experiences.