# Graded Unit 2: H48W35/011

Falkirk's Football Club Solution

Evaluation Stage

Name: Jose Fernandes

EC number: EC1637434

Date: 29/05/19

# Contents

# Outline of the assignment

The assignment given, outlined a system that had to hold information about a football club's weekly updated weekly, list of cooperated events as well as all information about the current league, which includes the table, match report, team and player details as well as a live match feed. To have access to this system, the subscribers need to first register for an account and then to login in the account, every time they wish to seek for information. This account had to be fully customizable to allow the subscribers to update their personal information. The system would also have two types of subscriber accounts, the partial and the platinum ones. The platinum subscription is a paid subscription, either paid monthly or yearly, that offers features not available to the partial subscription, such as live match feed for example. The payment system required to undertake the system payments was PayPal.

Additionally, there it was required an admin panel that would manage all the system's information which consists in create, update and delete any record in the system. The admin would also manage the users' records but with restrictions as he was not allowed to update certain personal details.

The football club given was Falkirk FC, playing in the second division of Scottish Football. This club is an important club in Scotland and its fan base is local to the Falkirk council and it is relatively large. For these reasons, the system had to meet the fan base high expectations in terms of design and performance. In terms of design, the system had to look simple and matching the football club's colours which in this case are mostly blue and white.

The solution built, to cover the wide range of potential subscribers, was a web application. This web application was a single page application that decreases the broadband required for subscribers to use it, especially mobile users. In terms of design, the design is mobile friendly, and its colour theme is based on the football club's main colours. The intent of choosing a web application is to allow a user to access the application from anytime, anywhere and on any device.

The web application has the admin and subscriber panels integrated together in the same application so that the administrators can also access the application easily. The authentication implemented in the web application was role-based, for the purposes of protecting unauthorized access to the administrator capabilities. The admin panel manages the users, news, events and the league data, which includes games, teams and players information, as well as its own personal details. The subscriber panel allows the user to update his information, upgrade his subscription and his type of account to admin and view the news, events and league information. The live match feed was also integrated and the admin, whenever is in the admin's match report page, will broadcast a live match feed that will be available to all subscribers in real time. In all, the required objectives were achieved by this web application.

## Strengths and weaknesses

The solution has revealed itself as a beneficial and resourceful solution which reflects what is required by a football club. This application has the potential to be extended to other types of applications. The architecture of Falkirk FC's web app is actually two distinct applications. It is divided into server application and its front-end Angular application. The separation of the application at the presentation OSI model layer, allows the server to communicate to any other type of applications (for example IoT or native applications) independently from the front-end application.

The Angular framework is based on the typescript, an object-oriented strongly typed language that is transpiled to javascript to run in the browser. This language has improved the development of the application by reducing the possible errors and by adding object-oriented features (that are non-existing in javascript) that increase the web application extensibility. The use of typescript produces robust applications that are highly extensible and readable.

The git version control system was essential to the solution's production. This version control system, alongside the ssh networking protocol, provided a safe option to control the source code versions, its remote storage in Github and its deployment in the Heroku server. This software has also provided a resource to safely commit changes to the source code, and even revert those changes if there is any kind of issue.

The server was built on top of Nodejs environment, which runs o javascript. The javascript language was not created to run on the backend of an application, even though it allows for a more flexible and dynamic type of programming. The use of this language in the server revealed to be error-prone and not as extensible as it would be in typescript. Due to this issue, a significant portion of the available time as literally wasted in productivity terms and other features could not be implemented, namely the front-end and back-end automated testing. The automated testing originally planned to be implemented in both back-end and front-end, was required a great deal of self-researching as well as time to implement it. For these reasons, the automated testing implementation was dropped from the project. The automated testing would

make the server application even more robust and extensible as errors would be checked for automatically, during the implementation of new features.

# Recommendations

As a result of the Falkirk's application architecture, there are many options to integrate many other types of applications.

The administrator panel could be improved by creating a desktop native application where the application's administrators would have an even better interface only focused on the management of the content. This interface can extend the existing capabilities of the current web-based admin panel and add even more capabilities requested in the future. This improvement would also remove the administrator's lazy loaded Angular module from the front-end application, reducing the number of files downloaded. These development changes would not affect the subscriber's experience has its panel would remain fully functional, only requiring minor changes.

IoT components could also be installed in the stadium to track in real time the conditions of the weather, the attendance or even artificial intelligence systems to match statics. All these systems could then be easily integrated into the server increasing the possibilities for subscribers and administrators.

In general, the player, team, match and match events models could be improved to hold more information and increase the quality of the current content. For instance, the player profile could have the date of birth and nationality to deliver more information and insight into the team in general. The team model could also be improved by adding the stadium so that in the match mode, the stadium hosting the match would be the home team. The match event model could also be improved to have the goal assistant to have a better insight match insight.

Additionally, the subscriber's module could have an additional section for player's tables of best scorers, best assistants, more yellow cards and more red cards.

# Modifications

The project's major modification was the implementation of the JavaScript language in the back-end server, instead of the typescript language which was specified in the planning. The typescript language offers object-oriented features, not available in native JavaScript, and it implicitly does type checking on the variables. If this language was used, the program errors would be easily spotted during compilation and the additional object-oriented features would allow for a more reusable and robust back-end server. The server's development proceeded was the planned and self-research was done to determine how exactly to implement typescript in the back end. The initial routers and controller were implemented in typescript but when the data models started to be implemented using the Mongoose library, some errors started to appear. During this stage of development, troubleshooting techniques were intensively used to try to create the data models as it was planned but the main issue become the available time, as it was taking longer than what was expected. The development of the data models was definitively possible but it would take longer than expected and due to the lack of experience implementing this language in the back-end, the decision was to implement JavaScript in the applications instead.

The other major modification was the implementation of the test logging technique instead of automated testing, in the front-end and back-end. Due to the loss of time when implementing the typescript in the server, further research on how to implement jest and jasmine libraries was cancelled. These testing frameworks were totally unfamiliar and a great deal of time was allocated in the planning to properly implement the required testing. As these frameworks could not be implemented, the testing was done by manually testing each possible error case hoping to detect all errors before production. The test logging testing was an important component to detect the most obvious errors but the automated testing is extremely more productive.

## Knowledge and skills

This project was essential to develop management and programming skills required to carry out large projects. The project management skills were improved due to issues that forced significant changes to the original plan. If these setbacks were not properly managed, they could have threatened the successful completion of the project as some of these setbacks required assertive and pragmatic decisions.

The programming skills were also improved due to the project's scale and to its complexity. The original plan laid down for the project had many intrinsic technical challenges that had to be effectively tackled. As a result of tackling most of the technical challenges, the existing programming skills, such as troubleshooting, object-oriented design and error handling were levelled up. Some new programming skills were also acquired such as asynchronous programming, thanks to the JavaScript's native promises and to the RxJs's library observables. The knowledge and experience in the typescript and javascript languages were also dramatically increased to meet the planned solution's expectations.

The back-end implementation, using typescript could have been successful if a framework, based on the language, was used instead. There are a few already mature typescript back-end frameworks available but the original plan was to produce a simple back-end typescript application. This problem could have been spotted if deeper research was done to identify the implementation's issues beforehand.