

**ASSIGNMENT COVER SHEET**

**Student Name Michael James, Martin Zuber, David Kelly**

**ID Number: B00019330**

**Course: Computer Science**

**Year: 1 of 1**

**Lecturer: Luke Raeside**

**Title of Assignment: Fantasy Hurling**

**Due Date: 10/5/15**

**Date Submitted: 10/5/15**

The material contained in this assignment is the author’s original work, except where work quoted is duly acknowledged in the text. No aspect of this assignment has been previously submitted for assessment in any other unit or course.

**Contents**

[**Introduction** 4](#_Toc417297464)

[**Project Description** 4](#_Toc417297465)

[**Project Specifics** 4](#_Toc417297466)

[**What We Will Look at in this document** 5](#_Toc417297467)

[**Review 1: Improving website design** 8](#_Toc417297468)

[**Review 2: The HTTP Protocol** 11](#_Toc417297469)

[**Review 3:** 13](#_Toc417297470)

[**Literature Review Conclusions (All reviews)** 24](#_Toc417297471)

[**Development Methodology** 25](#_Toc417297472)

[**Project Plan** 27](#_Toc417297473)

[**Use Cases** 28](#_Toc417297474)

[User checks current score 28](#_Toc417297475)

[User checks fixture 29](#_Toc417297476)

[User uses social media aspect 30](#_Toc417297477)

[User creates team 31](#_Toc417297478)

[User makes transfer 32](#_Toc417297479)

[**Sequence Diagrams for Use Cases** 33](#_Toc417297480)

[User checks score 33](#_Toc417297481)

[User checks fixtures 33](#_Toc417297482)

[User posts message 34](#_Toc417297483)

[User chooses team 34](#_Toc417297484)

[User makes a transfer 35](#_Toc417297485)

[**Wire framing** 36](#_Toc417297486)

[Login Page 36](#_Toc417297487)

[Registration Page 37](#_Toc417297488)

[Team statistics page 38](#_Toc417297489)

[Player transfer page 39](#_Toc417297490)

[Standings Page 40](#_Toc417297491)

[Social Media area 41](#_Toc417297492)

[**Work Flow Plan** 42](#_Toc417297493)

[**DESIGN and TECHNOLOGY details** 43](#_Toc417297494)

[**Implementation** 46](#_Toc417297495)

[**Deleting a player from the team** 56](#_Toc417297496)

[**Testing** 59](#_Toc417297497)

[**Conclusion** 62](#_Toc417297498)

[**References** 63](#_Toc417297499)

**Declaration**

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Degree of **Honours B.Sc. in Computer Science** in the Institute of Technology Blanchardstown, is entirely my own work except where otherwise stated, and has not been submitted for assessment for an academic purpose at this or any other academic institution other than in partial fulfilment of the requirements of that stated above.

**Abstract**

The goal of the project was to create an interactive and enjoyable fantasy hurling style application. It was to be for Hurling, but would be flexible in that the backend could easily be used for a different sport if and when it was needed. This made the project very modular. It was designed to be very intuitive and easy to use.

Some of the key features were the ability for the user to be able to create and maintain a fantasy hurling team, receive hurling new dynamically and interact with other users via an instant messaging style forum.

The result was an application that was very simplistic in its front end design to allow novice users and experienced users alike interact with all aspects of the application with ease. Administration of the application was simple using the administrator login to update and maintain all tables in the system. The application was very modular and could be expanded or used for a different sport very easily. It also tapped into the current phenomena of social media in its instant messaging style chat.

# **Introduction**

## **Project Description**

The project we are creating and that has been proposed in previous documentation is a fantasy hurling game. There will be a social media aspect to the application too. The game allows you create an account, then choose a squad of players from a list. You have a maximum budget and must stay under this. Each player is awarded a score each match based on real life GAA championship games. Each user’s team is then updated at the end of each game round. The main technologies used will be Java for the backend, html and JavaScript with some CSS for styling for the front end, and of course the database, which will be SQL.

There should be no real compatibility issues within the project, however a separate mobile version of the application could, at a future time be developed. There will be no special requirements for usage of the product. It will be web based as in it will be accessed entirely via the normal HTTP request in the browser, no special software needs to be installed, and no special hardware is needed for the application on the user’s side.

On the server side, we will host the application on Microsoft’s azure servers.

## **Project Specifics**

The objective of the project is to build a fantasy football hurling website, with an element of social media capabilities built in. The reasoning behind making this project is as follows. Anyone who plays fantasy football games knows that although they are quite interesting, a lot of the time is spent just logging in, checking your score, maybe making a change or two, and then logging out. The amount of time actually spent on the site is minimal, I feel this is a mistake in the design of these applications and games.

## **What We Will Look at in this document**

In this document we will provide some analysis and conceptual design of the product using UML diagrams and modelling techniques. We will look at the following:

* Comprehensive Use Case Diagrams
* Development Methodology
* Sequence Diagrams
* Class Diagram of the final product
* Activity diagrams
* User Interface Design
* Database table and schema design and description

**Main research questions**

The main questions of research in regards this project are how can we improve on similar apps that are already out there, how much can we find out about the game of hurling, can we find out how to get live figures through a RSS feed, is it possible to use real Logos or are there licencing issues, and of course what technologies will we implement in our project. We should also try research if there are other better versions of what we might be able to create out there, we need to find a gap in the market, and if one does not exist the project is doomed. A big area of research for all of us will be how to use Java server to our advantage as we plan on using this in the project.

**Justification and benefits to the project**

• There are no similar apps out there for hurling

• Hurling is a very well followed game

• The social media aspect can make it even more popular as outlined in the next section

• The instant messaging service would be unique to a fantasy gaming app

• The possibility of advertising revenues is very high

• Most importantly, hurling fans we have talked to want it!

**Main project Objectives**

* Stable working website
* Play a fantasy game
* Social media aspects to it
* User can register
* Ads can be placed on the site
* Instant messaging service
* Forum service
* Scores update using RSS feed
* Player score updated automatically on login
* Admin can log in to do maintenance

**Success Criteria**

* User should be able to register
* User should be able to select a team of 8 players
* User should be able to user instant messaging service to chat
* User can post on a forum
* Users points automatically updated each week
* User can chat with others near using a GPS system

**Deliverables:**

**Management Deliverables:**

Feasibility Study and Project plan: A feasibility study will be conducted so that a decision on the viability of the project can be made. This should contain an outline of the project along with the benefits, requirements and alternatives if any.

Analysis and design document: This document will contain all the requirements and functions that the Application should have. This will contain our UML diagrams which will plot how the user interacts with the software and what extra functions algorithms we might need.

Source code: We will use the Analysis and Design document to build the code for our software. This file should contain all code, images and installation files.

Testing Document: All tests performed on the software will contained in this. Testing will done in all stages of the application and be documented for further record.

**Technical Deliverables:**

* A clear and easy way for a user to navigate through and complete the team selection
* User login to access an account or set up an account on first use.
* User should be able to change a player once a week
* Administrator login to change variables without interfering with ongoing game
* Database to store account details.
* Real time update of player scoring using RSS feeds
* Instant message style chat built in
* Forum messages used for discussion of games
* User able to join leagues and check overall standing

**Literature Review**

The project required extensive research of literature. We each picked a topic and reviewed a relevant piece of literature. One topic was social media, another was back end development. We used the knowledge gained in these reviews to help us in the overall project.

# **Review 1: Improving website design**

Who did what?

Melody Y.Ivory and Marti A.Hearst University of California, Berkeley 2002 investigated what makes a good design to a website, and developed a model to help users implement these changes that would make the website more pleasing to the eye and in terms of usability.

Summary

Two students undertook a paper to investigate current methods of website design, their effect on productivity and how it can be improved though better website design principles. They undertook this as part of the Web Tango project. They aimed to help steer the average web site builder away from poor design principles and toward an automated quality checking tool and a grammar checking tool. They made these tools available online at “webtango.berkeley.edu”.

Melody Y.Ivory and Marti A.Hearst (2002) found that a website is a complex mix of text, links, elements and formatting, surmising that all these aspects affect a websites quality and usability. They came to the conclusion that these principles are important in thinking about website design to begin with.

The students came up with a table that created a way of calculating how many measures for each element on a web page were needed in order to come up with a design that is pleasing and easy to use. For example, they came to the conclusion that there are 31 separate measures that are important to think about in regards a text element including amount, size and complexity. The table also related to site architecture and the performance of the page, all elements discussed had a set of measures that were important to that particular element to optimize design.157 Measures in total were found.

Melody Y.Ivory and Marti A.Hearst (2002) then ran their crawler tool and used this to gather sample web pages. Initially it ran on the home page of a website and randomly selected pages at successive levels starting at that page and only selected informational pages ignoring advertisements or flash pages totally. The analysis tool then runs on these pages and retrieves the information on each element in conjunction with a site metrics computation tool, forming the table they discussed with the elements and their measures.

The students found there were three main principles to successfully designing a webpage; navigation design, graphic design and experience design. From these 3 main principles a hierarchical pyramid model was built, surmising that on the top level is the site architecture, while on the bottom are the actual site elements themselves. Web design literature and user studies were used to come up with the final model. A tool was then developed from this model that could compute 157 site level measures. The accuracy of this tool was tested on many websites and it was found to be 86% accurate on 154 measures.

The students performed three studies to try to predict page and site ratings. From these 3 they developed a simple prediction model. They called it the WebTango model. Firstly they drew up an analysis of 428 web pages and found expert reviews and ratings on these pages from PC Magazine’s top 100 sites. They called sites either rated or un-rated and set out to find a way to predict which category a sample site would fall under. They then computed 12 quantative measures related to page composition and design among other factors. They tried to see if they could predict with their model the page standings on this top 100. They found that 6 features were most important to design. The most prevalent were text cluster, reading complexity, and colour count and page size. They found that in rated sites these features needed to be tweaked a certain way to make the site very usable and stay in the rated group.

A second study was conducted and asked 6 website design experts to examine 1898 pages from the Webby awards winner’s websites. These pages were judged on certain criteria including content, structure, navigation and visual aesthetics. They broke these pages into three groups, “good”, “not good” and “poor”. They wanted to see if their model, using the measures they had created, could predict which group a site would fall into. Predictive accuracy was 67%.

The third and final study was to analyse over 5000 pages from 300 sites. They used all 157 measures from their model and again had 3 groups, “good”, “average” and “poor”. They used the model to try predict which group the web pages would fall under. It was proven to be accurate on a page level 96% of the time, and accurate on a site level over 60% of the time.

Melody Y.Ivory and Marti A.Hearst (2002) talked about their final task of applying the model they had built to website design. They took a sample of 15 web pages and made minor tweaks to these based off their model parameters. They asked 13 people to analyse the pages both before and after the adjustments had been applied to them using the model. They made findings that 10 out of 13 people preferred the web pages after they had used to model to make adjustments to it.

The students then analysed their final tool, the WebTango system. They analysed how the tool worked and how it can be applied to a website. The tool compared all 157 site level measures from the website, and then makes suggestions on how to improve the website and also gives links to example websites who are similar in type but have been designed to a higher level.

The major findings were in essence that a model can be developed by 2 students with little or no web design experience that can enhance the usability and visual aspects of a website very quickly by identifying areas that can be improved. It found that although some studies have found automated tools find it difficult to find problems with usability issues, the tool developed is considered a practical and useful solution to solve design issues early in the design phase of a websites development.

# **Review 2: The HTTP Protocol**

Literature Review

Hypertext Transfer Protocol is a stateless application layer protocol for communicating between distributed systems. (Podila 2013). HTTP is specified by RFC 2616 (Http 2015).

HTTP is connectionless, stateless and media independent protocol. A client sends a request and server answers with a response. After that the connection is closed and communication is forgotten on both sides. Current version of HTTP is 1.1.

HTTP is using Uniform Resource Identifiers (URI 2015) for resource requests.

**Generic URI format (HTTP Tutorial. 2015)**

URI = "http:" "//" host [“:" port] [abs\_path [“?" query]]

Example:

https://www.youtube.com/watch?v=BNgU-ZaF06w

**HTTP request (HTTP Tutorial. 2015)**

* A Request-line
* Zero of more header fields ending with CRLF
* An empty line ending with CRLF
* A message body (optional)

A Request Line generic syntax:

Request-Line = Method SP Request-URI SP HTTP-Version CRLF

Request methods:

* GET: asking server for a resource located at given URI
* HEAD: similar to GET request method, but only header is returned from server
* POST: asking to send a data in request body to server
* PUT: asking to replace a resource located at given URI with new data in request body
* DELETE: asking to delete a resource located at given URI
* CONNECT: asking to establish a tunnel to server identified by given URI
* OPTIONS: describes the communication options for resource at given URI
* TRACE: test the resource availability

**HTTP Request Simple exampl**e:

GET <http://www.itb.ie/> HTTP/1.1

User-Agent: Fiddler

Host: <www.itb.ie>

User-Agent and Host are some of many predefined header fields. Custom fields can be introduced as well. This request message doesn’t contain any message body.

**HTTP response (HTTP Tutorial. 2015)**

* A Status-line
* Zero of more header fields ending with CRLF
* An empty line ending with CRLF
* A message body (optional)

Status Line generic syntax:

Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF

Status Code:

It’s 3 digit integer. First digit defines the class of the status.

* 1xx : Informational
* 2xx : Success
* 3xx : Redirection
* 4xx : Client Error
* 5xx : Server Error
* **HTTP Response exampl**e:

HTTP/1.1 200 OK

Cache-Control: private

Content-Length: 14887

Content-Type: text/html

Server: Microsoft-IIS/7.5

Set-Cookie: ASPSESSIONIDQAADTRCR=KIIFFBMDCAOAICBNIDKKACIH; path=/

X-Powered-By: ASP.NET

Date: Sun, 05 Apr 2015 09:36:25 GMT

<! DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "<http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd>">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

Rest of the html omitted…

Example above is the response to <http://www.itb.ie> request. Response returned with status code 200 which indicates the success. After Status Line the various header fields follows. Then there is an empty line followed by message body. The message body contains actual html document of requested web page. Most of the html document is omitted in this example.

# **Review 3:**

The effect of fantasy sport participation through social media

Sport spectatorship is one of the largest forms of leisure behavior in the world [1]. Large numbers of sports fans attend sporting events frequently and follow their sport ubiquitously through various mediums, TV, radio, newspapers, and magazines. With advancements in Internet technologies, fans are redirecting their consumption of sport to online platforms. Sports fans can now consume news and content via applications and social media on mobile devices. Fans can connect with each other through tweets, texts, blogs and Facebook.

Fantasy sport participation is not a new phenomenon; the history of fantasy sport is a topic of debate. In America the formation of a fantasy baseball league in the 1960’s by a man called Bill Gamson is considered to be the first documented account of a fantasy league [2].

Today the exponential growth of fantasy sports participation is fuelled by the increased televised broadcasting of sport, accessibility of sport and statistics on the Internet and the advent of social media as a communication outlet for fans worldwide, because of this, fantasy sport platforms are now formally aligned with many official major sporting leagues globally.

Fantasy sport provides engaged fans a unique and personal opportunity to be involved while also building the interest and knowledge of new participants with lower prior involvement. Further, fantasy sport provides additional social interaction between players. It can serve as an education tool about the sport, teams and players and can also serve to increase the commitment and interaction of consumers with a sport.

2.2 Existing Trends

In North America and Canada the fantasy sports industry has a body that monitors and provides information on all fantasy sports websites known as the Fantasy Sports Trade Association (FSTA) [3]. The FSTA released figures on the number of participants in 2011, more than 35.9 million users are involved in some type of fantasy sport. The FSTA then surveyed the group to find out how many users participate using mobile devices. The survey found that 30% (12,000,000) of users are engaging through mobile devices. Further, of the 12,000,000 users that play through mobile devices 14% (1,680,000) use social media to discuss their fantasy games.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Members | Mobile Users | Social Media Users |
| BPL | > 3,000,000 | 900,000 | 125,000 |
| FSTA | >35,900,000 | 12,000,000 | 1,680,000 |

The Barclays Premier League (BPL) claims to have the biggest football fantasy game in the world [4]. The Fantasy Premier League game has over 3,000,000 participants. If we apply the same calculations from FSTA to the Fantasy Premier League game we can get an estimate value of over 125,000 users that discuss their fantasy football game via social media.

Reason for Literature Review

This literature review is being conducted to ensure a thorough understanding of the connection between social media communities and fantasy gaming. I am currently working on a project that is combining a fantasy-hurling platform that enables social media capabilities.

Upon completion of this literature review I hope to identify potential areas of difficulty that may be unforeseen without specific research in to the topic of social media and fantasy gaming.

**3.0 Literature Review**

Joris Drayer, Stephen L. Shapiro, Brendan Dwyer, Alan L. Morse, Joel White. (2010).

“The effects of fantasy football participation on NFL (National Football League) consumption: A qualitative analysis”. *Sport Management Review, 13*, 129-141. [5]

Drayer, Shapiro, Dwyer, Morse and White proposed three areas of interaction that may be affected for fantasy football users and non-fantasy football users.

1. Identification and loyalty
2. Consumption
3. Attitudes and behavior

3.1 Identification and loyalty

To conduct a test on the fantasy gaming NFL fans, the authors must first define a norm. A fan that does not participate in fantasy football and the attributes of one such fan. A normal fan is considered to have allegiance to one team. The normal fan supports the team and the players that play for their team and no one else.

The authors expected gamers to have no loyalty or identification to a certain team, but rather have identification to a player. This was not the case, gamers still identified themselves with a team but their affiliation to other players via the fantasy game was just an extension.

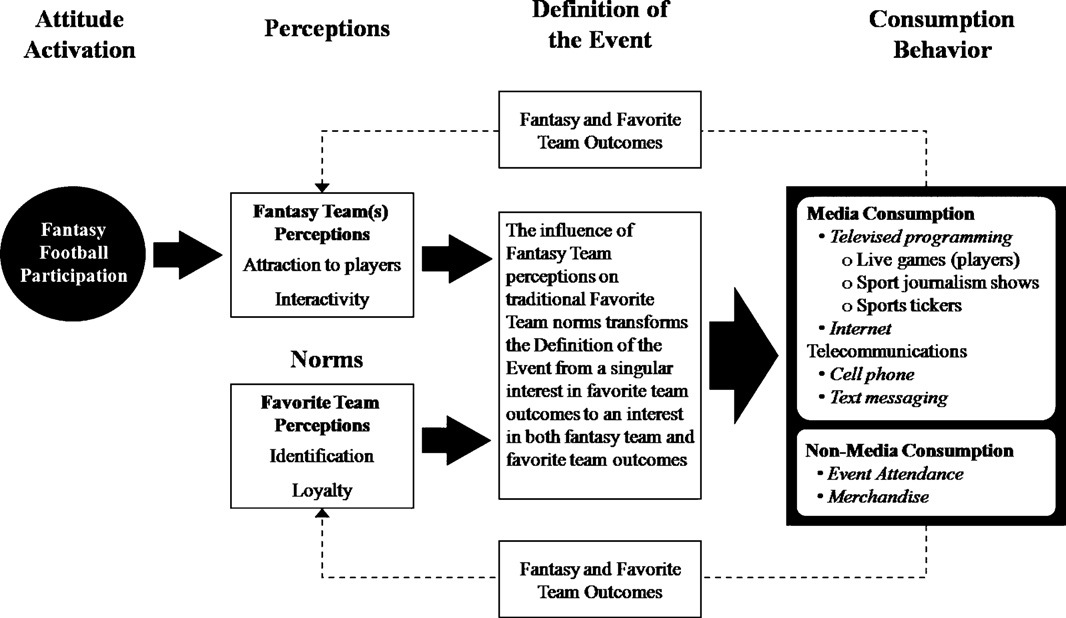
Those interviewed stated that their loyalty to a fantasy player ended with the season.

Only 2 people of the 13 interviewed said that they carried affinity for a player in to the next season and rooted for them if their favorite team wasn’t playing.

3.2 Consumption

To conduct this test the authors once again had to define a norm. A normal fan is one that only consumes information about their favorite team.

The authors expected gamers to consume NFL information in a different manor. Having affiliation with many players in many different teams’ leads to greater consumption of information.



Fantasy gamers have an increased media consumption of the NFL, spending time on the Internet and watching ESPN to gain information on starting rosters, injured players and to check which teams are playing in game weeks. Of those surveyed participants admitted to checking their mobile device and going on to the Internet every 10 minutes while a game is on. Some gamers paid for subscription sites to get informed advice on their fantasy teams. Fantasy football participants found themselves watching more matches on game weeks because they had numerous players playing in different games. When asked if they would watch as many games if not playing the fantasy football game, the answer was no.

No change was noted in the form of consumerism. Only one person said they would buy the jersey of a player because she began to like him in her fantasy team.

3.3 Attitudes and behavior

The control norm for this test was that of a fan, who regardless of their favorite teams result would support them.

The results found in the survey suggested that a fantasy football player would have allegiance for a team but if that team lost, the fantasy football player would adjust their attitude towards a real team to suit their fantasy game. When the 13 fantasy gamers were asked about their behavior towards an opposing player, 2 gamers admitted to supporting opposing players of their favorite team.

Those interviewed stated that if their fantasy football team began to lose and become uncompetitive they would lose interest and stop updating their team towards the end of the season.

3.4 Research Paper Conclusion

The research results show that fantasy gaming participation really does have an effect on how fans, consume, interact, behave and identify themselves with teams. A non-fantasy gamer does not consume as much information over a broad scale but rather focuses solely on their favorite team. With fantasy gamers this is not the case. Fantasy gamers spend more time online, on blogs, sports news websites, social media, TV, magazines trying to get information on their fantasy team. Fantasy gamers tend to be more fickle in their allegiance to a team and player, favoring and supporting teams and players that can generate points for them in their fantasy league.

Kurt Komaromi, Scott Erickson (2011).

“Using Social Media to Build Community”, *Competition Forum,* *October 27-29*, 325-333. [6]

Komaromi and Erickson follow a template laid out by Culnan, McHugh and Zubillaga that assesses the effectiveness of an organizations social media strategy. Effective implementation of a social media strategy is one that attracts an audience enabling them to engage and interact with both the organization and each other.

Although the study is not conducted on a sports category, the essential requirements for a successful social media community apply in all communities. The study instead researches 18 insurance firms over a two-month period in 2011. The authors then present specific results for three firms that display a range of social media outlets to support a community.

The three insurance companies presented are, Progressive, Liberty Mutual and State Farm. The study begins by analyzing the company’s website homepages. The homepages all display a range of tools that allow customers to interact by getting quotes, mobile apps, price comparisons and calculations for various insurance plans. What is important on all these homepages is the prominence of social media resources. All three websites have social media icons to connect via Twitter, Facebook and YouTube. Liberty Mutual is the only site that has a LinkedIn page that allows customer access. State Farm has an additional social media outlet through Flickr.

Metrics for each homepage visit can be seen in the graph below. Progressive has the highest unique visit rate at over 4 million visits per month. State Farm is second with 2.5 million and Liberty Mutual is third with 1 million unique visits.



3.5 Facebook

Facebook is the first social media outlet covered in the findings. All three insurance companies are present on Facebook.

Progressive have a Facebook page that primarily provides information about the company, offers and customer feedback. Progressive also have a current marketing campaign that attracts the most followers of the three companies Facebook accounts. “Flo” is the name of that marketing campaign and has a Facebook account that is updated daily and has interactive videos and games to engage customers. At the time of research, Progressive has a following of over 2.5 million on their Facebook accounts.

The Liberty Mutual Facebook page is similar to Progressive, They have an about section to provide information about the company. Further Liberty Mutual has videos, comments, polls and games on their page to engage customers. Their Facebook wall has daily posts from employees and brokers that can be interacted with by visitors via likes and comments.

State Farm’s Facebook page offers a diverse content including offers, promotions, and sponsorships for social causes, games to rate drivers and apps that can be played with friends. The site also has a driving game application to complete a safe driving discount offer. Their wall has many comments both by State Farm and visitors.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Progressive | Liberty Mutual | State Farm |
| Facebook | ~2,500,000 followers | ~10,000 followers | ~90,000 followers |

Although Progressive have an advantage with a larger hold on the market, the evidence is clear that an updated and well-maintained Facebook page that interacts with visitors gathers more followers. State Farm has a large number of followers because they engage their visitors with apps, polls, videos and respond to comments daily.

3.6 Twitter

Twitter is the second social media outlet researched in the report. All three insurance companies have an active twitter presence.

Progressive utilize twitter to connect with customers with posts daily and responding to complaints, questions and criticism. Progressives Twitter account post basic answers to very common customer questions in a timely basis.

The Liberty Mutual Twitter account is active once or twice a day. Their Twitter account is used to promote news and answer questions from customers. Liberty Mutual also use their Twitter account to promote sponsorship activities and encouraging followers to attend social events.

State Farm populates their Twitter feed with posts daily, ranging from comments, questions, complaints, and re-tweets from other third party sources. The emphasis is on responding to customer queries. State Farm is more concerned with customer comments and feedback than Liberty Mutual and Progressive. State Farm shares a lot of content from their other social media sites through twitter, connecting their Facebook, YouTube and Flickr accounts.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Progressive | Liberty Mutual | State Farm |
| Twitter | ~7000 followers | ~2500 followers | ~30,000 followers |

3.7 YouTube

YouTube is the third and final social media outlet researched in this paper. All three insurance companies again utilize YouTube as a connection tool to their customers.

Progressive don’t only use their YouTube account as a video player but also implement a quote app link and link to their other social media outlets. The site has adverts, promotions and visitor uploads. Progressive had 97 videos uploaded to their account at the time of research and the comments section on their YouTube channel is very active with over 400 comments.

Liberty Mutual has 31 videos uploaded to their YouTube account, providing both advertising and more in-depth information. Liberty Mutual does not update or maintain their

YouTube account frequently with only a hand full of comments on their account page.

State Farm has the most amount of content on their YouTube account, with over 180 videos at the time of research. State Farms videos range from adverts, promotion offers, to custom content. Fail Fridays is a popular video series that compares customer reasons for claiming insurance. State Farm also provides instructional videos to prevent accidents. Comments are extensive across all their videos and a find-an-agent app is available.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Progressive | Liberty Mutual | State Farm |
| YouTube | ~2,000,000 followers | ~250,000 followers | ~3,000,000 followers |

3.8 Research Paper Conclusion

Social media connections to customers are of huge importance to building a community around a product. The statistical analysis of three insurance firms provides evidence of such a statement. State Farm update their social media accounts more frequently than the other two firms as a result the community surrounding State Farm is larger and engages more with the company. State Farm makes it important to respond and interact with followers daily and update followers with relevant information regarding their industry and company.

**4.0 Research Paper Connection**

Although the topics in both research papers are very different, the ubiquitous nature of social media is the connection. Taking advantage of social media to connect fantasy sport gamers and allow them to share their information is easier said than done. In order to understand the best methods of engaging fantasy sport fans and encourage them to use social media as a platform to create and interact with a community is the problem.

The first paper reviewed states that fantasy sports gamers clearly consume information at a higher level than that of a normal spectator. Fantasy sports gamers gather more information regarding a sport through magazines, TV, radio, blogs and online. A part of making the transition to social media is by creating a community where user content is updated and fans can interact with one another, sharing information, tactics, statistics and analysis.

The second paper reviewed contains data and an analysis of social media communities that already exist. This data gives an insight in to how a social media community works.

State Farm take advantage of their social media presence more than the other two firms. The number of followers proves that maintaining a regular social media presence attracts followers. Having a Facebook page and Twitter account is essential to connecting with an online community. All three firms in the research paper use these social media outlets regularly to post comments and just as importantly to receive questions, input and criticism.

**5.0 Gaps in Research Papers**

After doing research external of this literary review I would like to draw attention to some gaps that I feel exist in both research papers. Although the connection between the two papers is very real and an outlet for fantasy sport gamer discussion can be managed in a social media platform, questions must be asked about the data analyzed.

The research paper by, Joris Drayer, Stephen L. Shapiro, Brendan Dwyer, Alan L. Morse, Joel White. **“The effects of fantasy football participation on NFL (National Football League) consumption: A qualitative analysis”** is a research paper about fantasy football in America and not sports globally, including hurling. Questions about a fantasy sports player consumption of information must be asked in order to get a complete view of the fantasy sport industry.

* Do all fantasy sports players consume more information about their sport than that of a normal fan?
* Do fantasy sports games accommodate gamer interaction?

The survey documented in the research paper reports on only 13 test users. The information could be incorrect or corrupted by such a small number of test users.

Kurt Komaromi, Scott Erickson. **“Using Social Media to Build Community”** researched the social communities surrounding three large insurance firms in America. The data gathered in this research paper is informative and formidable but questions still exist in the form below.

Can the social community of an insurance firm be relevant to that of a fantasy sport community?

* How do insurance customers differ from fantasy sports gamers?
* Have all forms of social media been researched?

**6.0 Literature Review Conclusion**

It appears that fantasy sports gamers may consume information and data of their sport on a different level to that of a normal fan. Fantasy sports fans follow a greater number of players and teams connected to their fantasy game. A platform to cater for an online community of gamers could exist to allow gamers to share information with fellow gamers. Fantasy gamers spend more time on mobile devices during games and between game weeks researching real life player and team data.

Fantasy players can still identify themselves normally with one team but can also extend their identification to a number of other teams during the season. Gamers however have a tendency to lose interest in their fantasy games if their fantasy team is performing poorly or their real life favorite team is performing well.

Social media communities flourish when content is updated regularly and interaction between community members is high. Building a social community takes many platforms through YouTube, Facebook and Twitter. Connecting social media outlets together such as linking videos on Twitter to videos on YouTube engages the community to interact within itself.

Having platforms high in content produced by community such as comments and questions creates a larger following. Users tend not to follow social media outlets that rarely post content. The insurance company with the smallest amount of content on social media platforms had the fewest number of followers. The insurance company that had the largest amount of content, videos, posts, tweets and applications had the largest number of followers. The connection between a maintained up to date social media presence directly relates to the number of followers in the community.

# **Literature Review Conclusions (All reviews)**

All three reviews were useful in research towards the eventual project itself. The reviews were of both technical and observational variety. The reviews themselves gave us fresh impetus in working on the project as they seemed to back up the idea that social media in this form would be something people would be interested in. The HTTP research was obviously vital to the project also as the front end heavily relies on these concepts.

## **Development Methodology**

In this section we are going to discuss the methodology of software development life circle (SDLC) we chose to use to develop this project. After initial research we decided for Prototyping SDLC.

**Prototyping SDLC, what is it basically?**

In this approach the development team implements a ‘sample’ which have only very limited functionality of proposed project and show it to the customer. Customer provides the suggestions of improvements and finally the development team implements these. This circle then repeated itself numerous times until the project is fully functional, tested and accepted by customer. This approach differs from other methodologies by avoiding doing the ‘big design in advance’ followed by implementation, testing and deployment phase. The project is rather developed by mutating the prototype with numerous design, implementation testing and deployment phases until the final product is build ***[1]***.

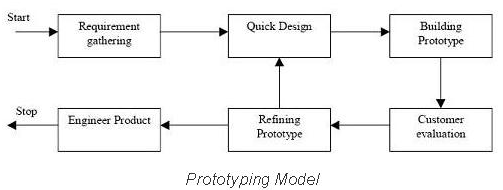


Figure 1 Prototyping SDLC work flow

**Why did we decided on this approach?**

We decided for prototyping approach for various reasons

* We wanted to avoid ‘big design in advance’. To design web application in advance it requires experience. Only experienced developers who already worked on projects with similar functionality and scope are able to do that. If ‘an amateur’ tries to design the application he will realize during the implementation phase that his design has flaws and redesign is required. This will lead to lose of work hours and any codebase the developer had already implemented. Basically you need an architect to design the house. We amateurs. We never build web application of this scope. With prototyping approach we be able to redesign the sample as many times as needed.
* We will learn with each prototype iteration. In our case we first build a simple but working sample build from prototypes. Data store prototype (relational database layer), data model prototype (objects to data mapping layer), controller prototype (business logic layer), and the view prototype (front end client layer). We make sure it all works together. Then we pick one proposed functionality and implement it whole way down through all the layers. And we learn from it. Implementation of next functionality will be easier and we get more productive over time. Hopefully after various iterations we will be able to call ourselves ‘the web developers’.
* Time restrictions. According to our project plan the first working prototype delivery is due to 14.February 2015. But we have been notified (by email 14.November 2014) that we must deliver the ‘working prototype’ due to 19. December 2014. According to our plan this is a two months ahead. Being still in research phase of project plan we assessed that only feasible approach under these circumstances is prototyping SLDC.

## **Project Plan**

*In this section we are going to discuss how we planned to develop this project.*

**What are we developing?**

Fantasy hurling project is basically fantasy sport game. It’s rich web application not a static web page. As a player manages his fantasy hurling team a content dynamically changes. At the same time it must have a persistence capabilities. Basically a changes player made must be stored somehow and not get lost when player finishes to play. Data must outlive the application runtime.

*Basically we will need to have three main layers.*

* Front-End (code running in browser delivered by initial http request from web server)
* Back-End (code running on server listening and answering to http requests)
* Persistence Storage (relational database server listening and answering CRUD queries)

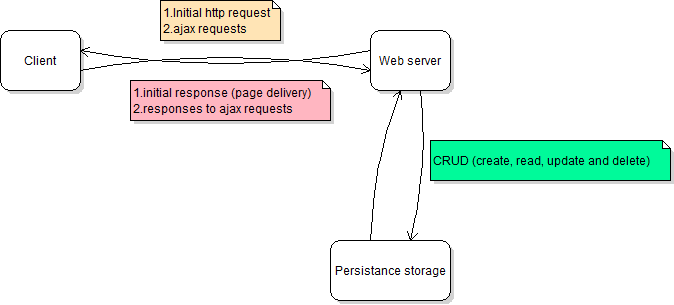


Figure 2 Web Application general design

## **Work Flow Plan**

* Due to change of SDLC methodology all DEVELOPMENT and QUALITY ASSURANCE items are merged into PROTOTYPING ITERATIONS item.
* All DESIGN items are reduced to PROTOTYPE DESIGNS (user interface prototype design etc.)

**Individual Task Assignment?**

**David Kelly**

* Front-End design ***responsibility*** (wire frames, photo shop, HTML5, CSS3)
* Documentation involvement
* Testing involvement

**Michael James**

* Front-End development ***responsibility*** (HTML5, CSS3, vanilla JavaScript, JQuery, JQuery UI)
* Documentation involvement
* Testing involvement

**Martin Zuber**

* Back-End ***responsibility***(MySQL database, .NET Web API 2, Entity Framework ORM mapper, Azure Deployment)
* Documentation involvement
* Testing involvement

## **DESIGN and TECHNOLOGY details**

## **Use Cases**

### User checks current score

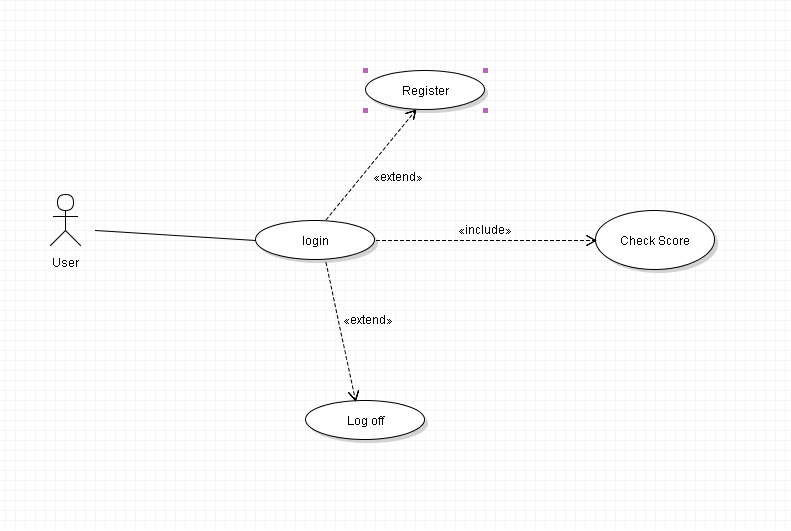
Use case specification:

1: User logs in

1a: User logs off

1b: User not registered and is prompted to register

2: User checks their current score



### User checks fixture

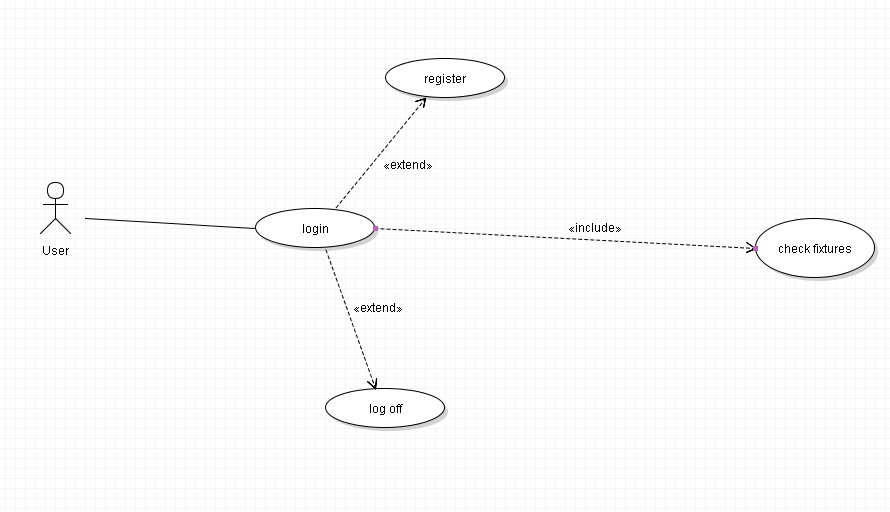
Use case specification:

1: User logs in

1a: User logs off

1b: User not registered and is prompted to register

2: User checks fixtures coming up



### User uses social media aspect

Use case specification:

1: User logs in

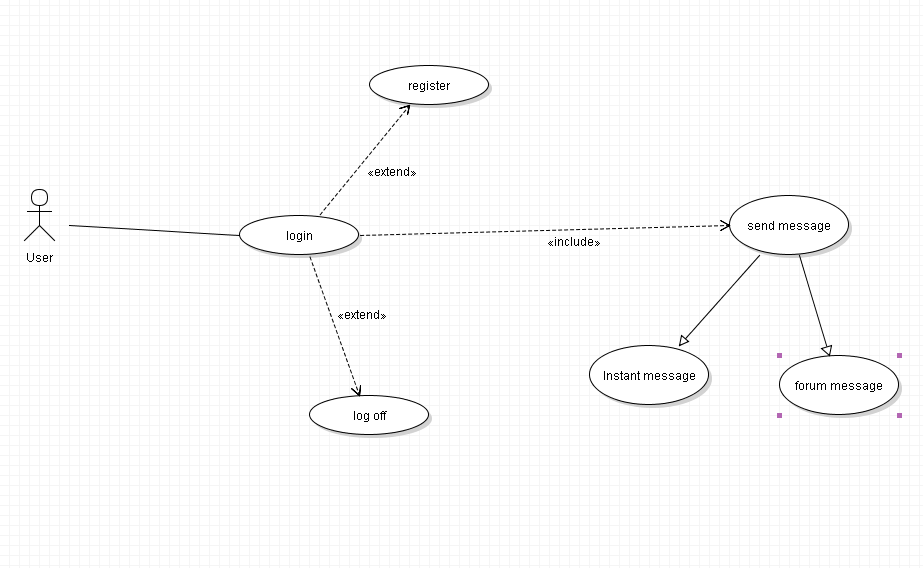
1a: User logs off

1b: User not registered and is prompted to register

2: User sends message

2a: User posts in the forum

2b: User sends an instant message



### User creates team

Use case specification:

1: User logs in

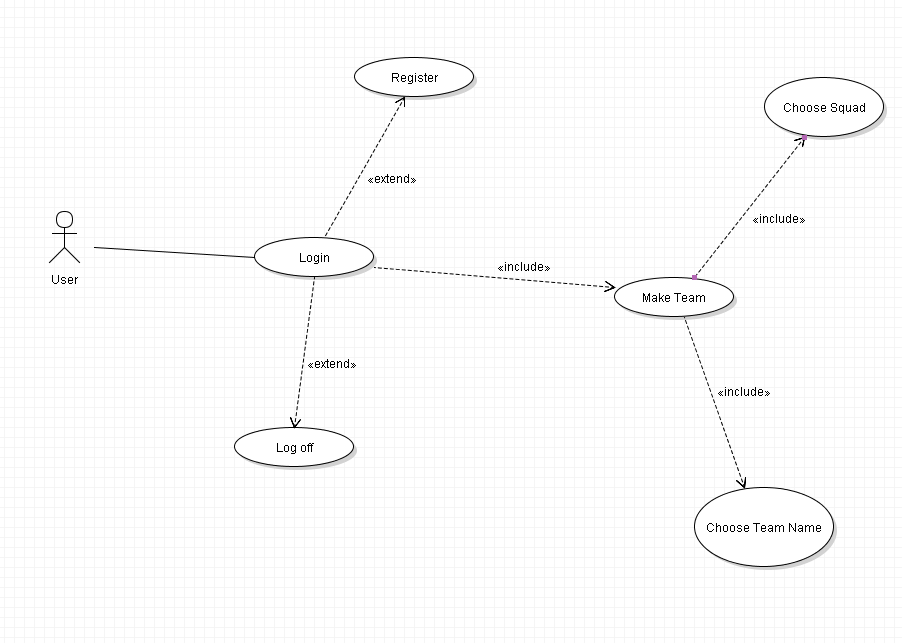
1a: User logs off

1b: User not registered and is prompted to register

2: User makes team

3: User chooses squad

4: User chooses team name



### User makes transfer

Use case specification:

1: User logs in

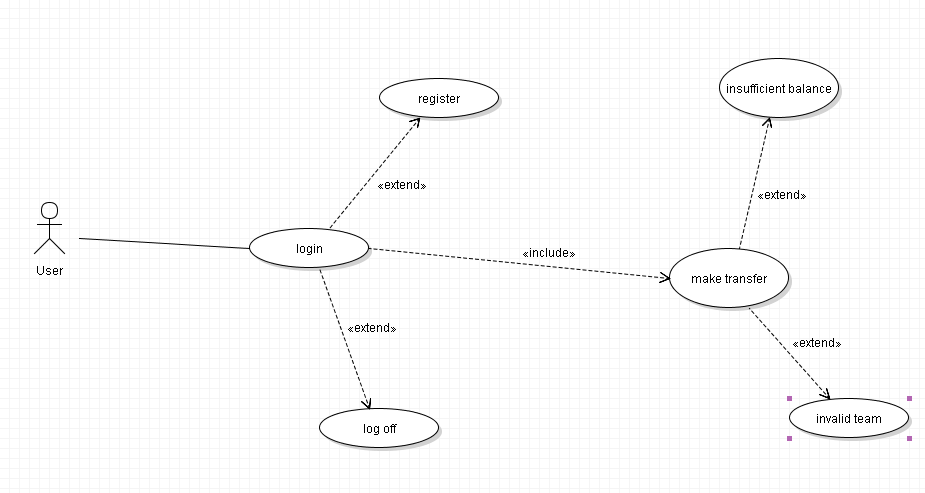
1a: User logs off

1b: User not registered and is prompted to register

2: User makes transfer

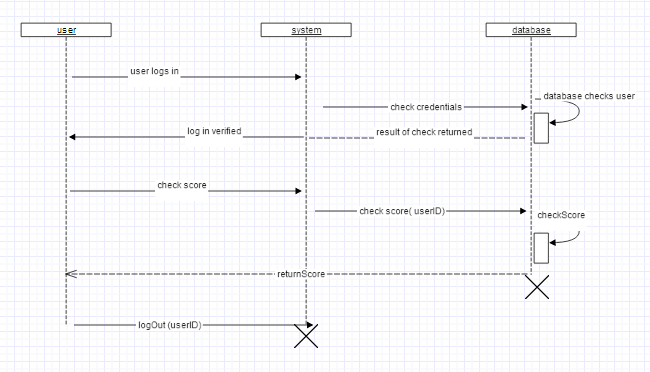
2a: User has not enough funds to make transfer

2b: User selects and invalid team choice



## **Sequence Diagrams for Use Cases**

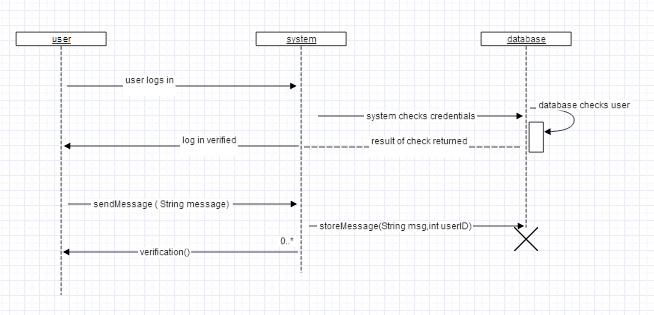
### User checks score



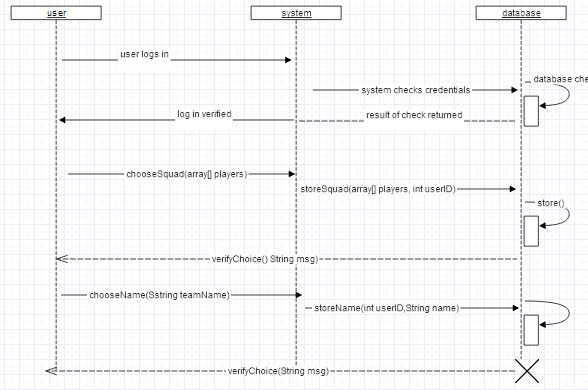
### User checks fixtures



### User posts message



### User chooses team

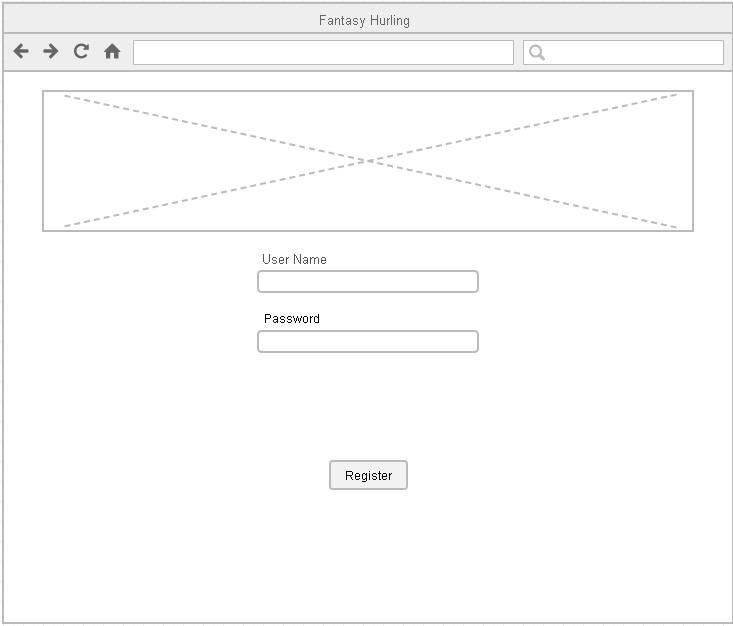


### User makes a transfer



## **Wire framing**

### Login Page



On this page the user can log into the system. They enter a username and password and are logged in. There is an image also.

Elements needed:

* 2 Text boxes
* One button
* One image

### Registration Page

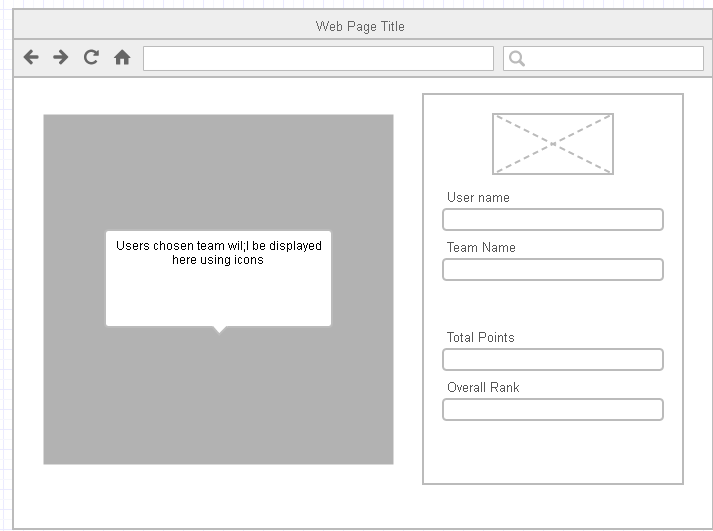


On this page the user can register if they have not already.

Elements needed:

* 5 Text boxes
* One button
* One image

### Team statistics page



On this page the user can view their team’s current standing and overall points. It shows the user name and team name and the team the user has selected on the left

Elements needed:

* One image
* Team box made up of images and text
* Some basic Divs to show some information based on the user

### Player transfer page

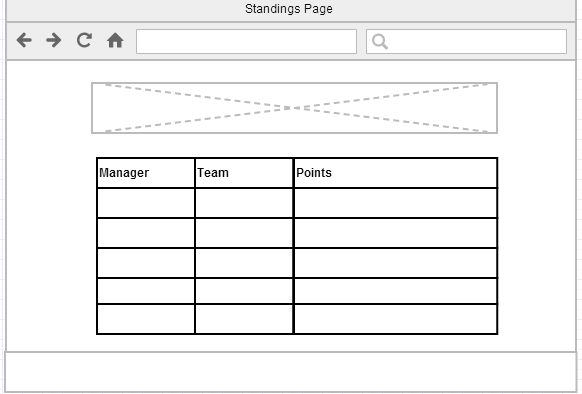


On this page the user can make transfers if they want to. They can search through the database of players, remove a player from their team, and see player score and value. Nothing is confirmed till the user clicks the add player button.

Elements needed:

* One combo box
* Basic Div. to show stats about the player
* One image to show player picture
* One button to confirm change
* A box on the left with all the players selectable and removable from the players team

### Standings Page

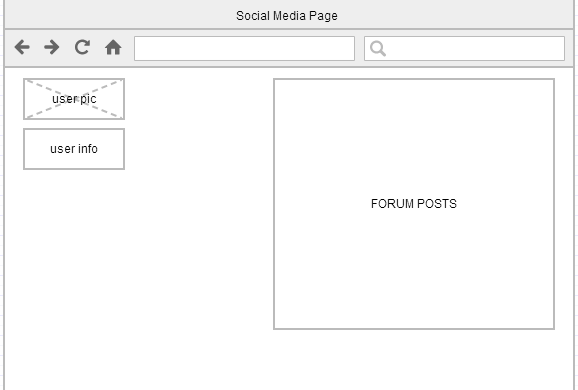


On this page the player can view the top players in the game by points scored. He may be in this list, it will be a simple SQL query displayed in a HTML table.

Elements Needed:

* HTML table
* One image
* Header and footer ECT

### Social Media area

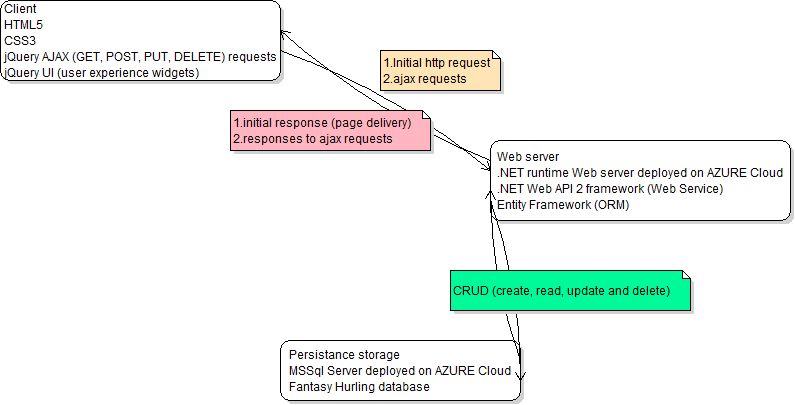


On this main social media page the user can view all recent forum posts and has some information about their profile displayed also, when the user clicks a post, which will be displayed as a title with a link they are brought to a separate page with that post and all replies in it. Here the user can reply to the post also.



Elements needed:

* One image
* One text area
* An area to store forum posts
* A button to post the message



*Figure 3 Web Application in detail*

**GIT and Github**

We are developing this application using Git distributed revision control system.

Fantasy hurling upstream repository is available at:

<https://github.com/Michaelcj10/Fantasy_Hurling>

Fantasy hurling repository clone is maintained and available at:

<https://github.com/zubidlo/group_project>

Web API prototype upstream repository is available at:

<https://github.com/zubidlo/FanHurApi>

These two repositories are going to get merged in later state of development.

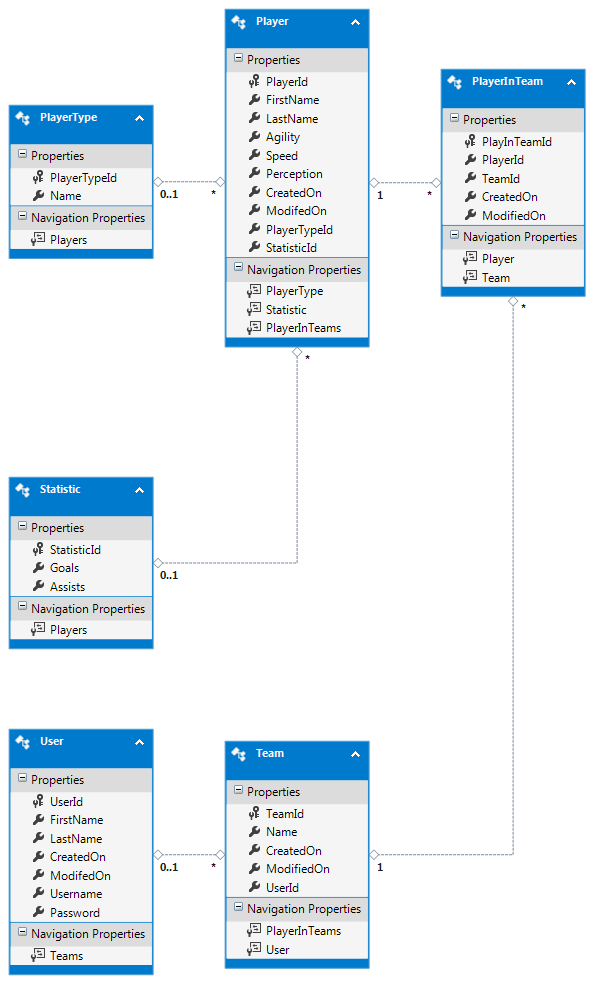
**Front-End**

First front end prototype is simple web page using HTML5 ***[2]*** and CSS3 ***[3].***AJAX ***[4]*** request are made through simple web forms using JQuery ***[5].*** In prototyping iteration this web page will gradually turned to user friendly application using JQuery UI ***[6]*** widgets and methods.

Front-End development will be done using IDE. Some recommended options are Sublime Text 2 ***[7]*** or Aptana Studio ***[8].***

**Data Store**

We store Fantasy Hurling data in relational database tables. Initial prototype is design to mimic football like sport team and player statistics and it will gradually mutate into final ‘hurling’ state during prototyping iterations.



*Figure 4 Fantasy Hurling database prototype*

Database will be deployed on Azure Data Store ***[9].*** At the end of development data store gets migrated to Azure Europe North Datacenter located in Dublin ***[10].***

**Back End**

Extensive research was done on web application back end development and we decided to build Web API Service ***[11].*** Initial idea was to use Node.js framework ***[12]*** to build the API, but after additional research a decision was made to use ASP.NET Web API ***[13]*** technology. Some of the reasons for this change include:

* Complexity of task versus maturity of Node.js technology
* Potential lack of documentation if a problem in code arises
* Lack of quality connectors to relational databases, Mongo DB (json based object database) is preferred with Node.js
* Additional frameworks to learn building Web API (express framework and more)
* Visual Studio 2013 Community edition free for use now
* 10 web servers free on Azure Cloud with Visual Studio installation
* Visual Studio fully implements Azure storage and web server deployment (on click in a menu and application and database is up on cloud)
* Comprehensive documentation and tutorials for all required frameworks and technologies.
* Technical support and huge community of developers if a coding problem arise.

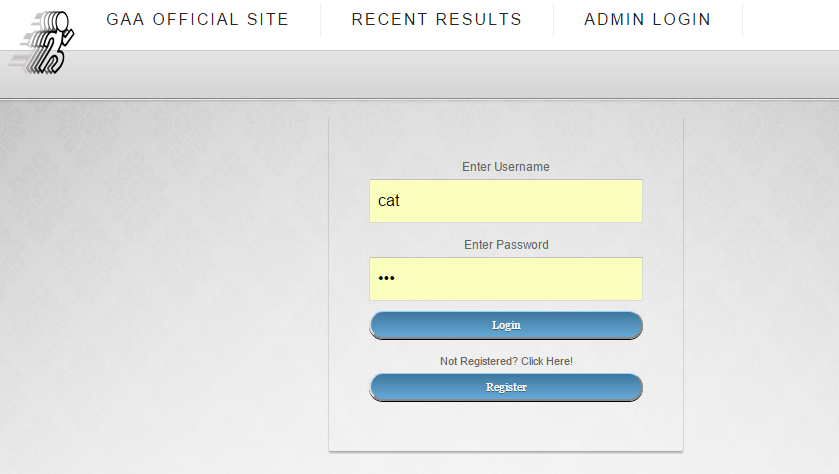
I must add here that we expect a usage of additional tools and technologies during the project development as a need arises.

## **Implementation**

**Front end**

There were some key elements to creating the front end. A series of simple but cleanly designed web pages were built. JQuery UI widgets were used, and JavaScript was used to connect the front end to the back end database. This was done using an API that was built by one of the team members in C#. We won’t go into this now as it will be explained in detail later. Navigation on the site was made as simple as possible and there were help icons on each page to explain how the user could interact with the system. We will look at the site in detail now and fragments of code will be shown and explained.

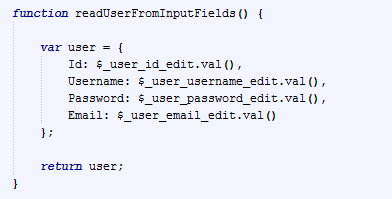
**Login and Register**



The user needed a password and username to login and interact with their team. The login page was kept minimal and straightforward to understand. The following is an example of the method that would log the user in if that username and password was correct. A URL was specified with the user ID the user had entered in the text field as their own and an AJAX request retrieved that users details if he existed. If he did exist, it retrieved the password associated with that account and checked it against the password the user entered, if it matched, the user was logged in, if not, a message prompted the user that he needed to re-enter the details.



If the user did not have a login, he could register to create one. He had to enter a username, password and a valid email address. The API would not allow duplicate usernames so long as it did not already exist, the account was created. A simple function was needed to add a new user to the account. The information to put into the database via the AJAX POST request was retrieved and stored in an array. This information was what the user had entered In the HTML fields.



The function to add the user into the database via the API was a simple POST request using the JSON data.



**Session Storage**

With local storage, web applications can store data locally within the user's browser.

Before HTML5, application data had to be stored in cookies, included in every server request. Local storage is more secure, and large amounts of data can be stored locally, without affecting website performance.

Unlike cookies, the storage limit is far larger (at least 5MB) and information is never transferred to the server.

Local storage is per domain. All pages, from one domain, can store and access the same data.

Session storage was used largely in this application. The bonus to the session storage was that it got rid of the need to pass variables between pages in the style of PHP which made accessing variables such as the username very easy.

Example of setting is below, we set our current user ID to be called “id” in session storage. We could not access this any time we wanted across the entire site with a very simply command.

Set session storage

sessionStorage.setItem ("id", id);

Get session storage value

Var user = sessionStorage.getItem ("id");

Session storage could be cleared at the end of the session using a simple command.

sessionStorage.clear ();

The user’s players in his team were stored in this manner also which allowed them being interacted with very easily. For example when the players in the users team were being retrieved each player was assigned a session variable of the data retrieved for player position, so the first player out was called “1” as his player position is 1. The value for this name of 1, was his player ID value. The entire team could be stored this way in session storage and then accessed at any time.

sessionStorage.setItem ("" + object.PositionId, "" + object.Id);

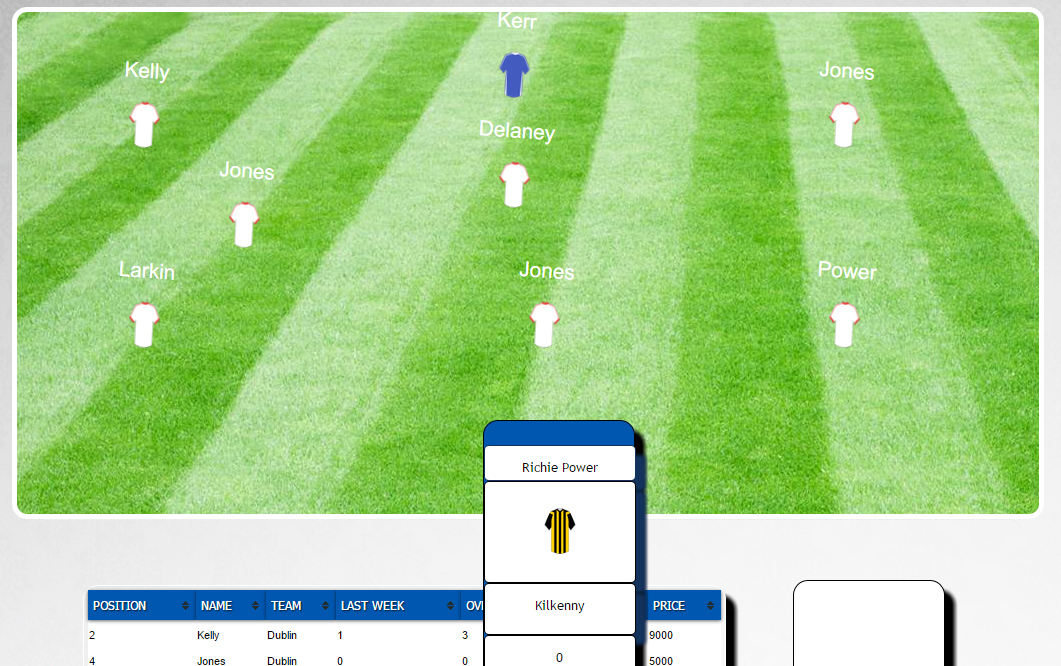
**Creating a new Team for the user**

When the user navigates to either the view team page or the transfer’s page, the application checks if a team exists with that user ID. If the team exists then nothing extra happens and the application acts as normal. If it does not exist, a new team is created for the user. The name is simply the username and the user ID combined. The type of request to do this is a POST request. The team ID is then stored in session storage.

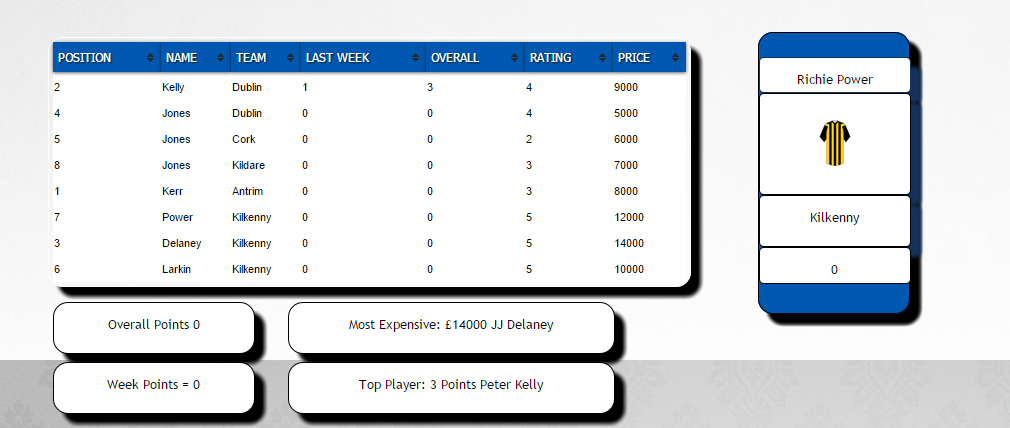


**How to view the team**

The user could view his team using the team view page. In here there was a graphical representation of his team using icons which were movable. Each time a user hovered over a player a baseball style card would pop up showing details about that player.

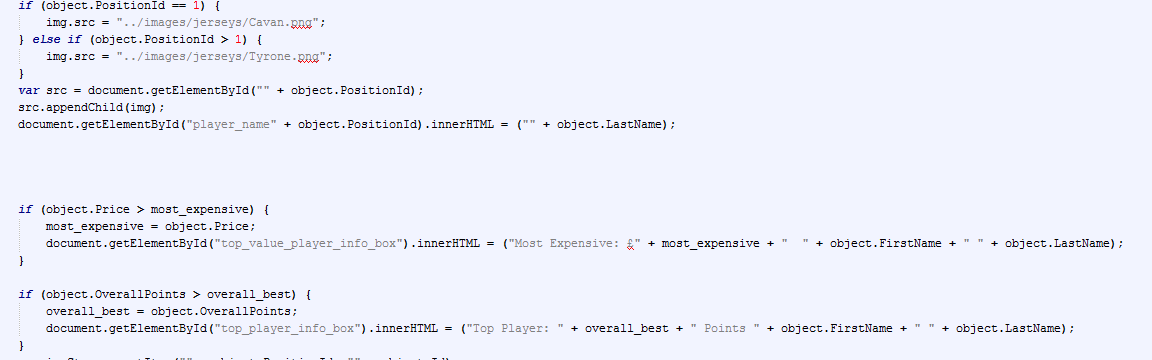


More detailed stats about the team were also shown in the form of a sortable table and some widgets showing information such as the top scorer or most expensive player.



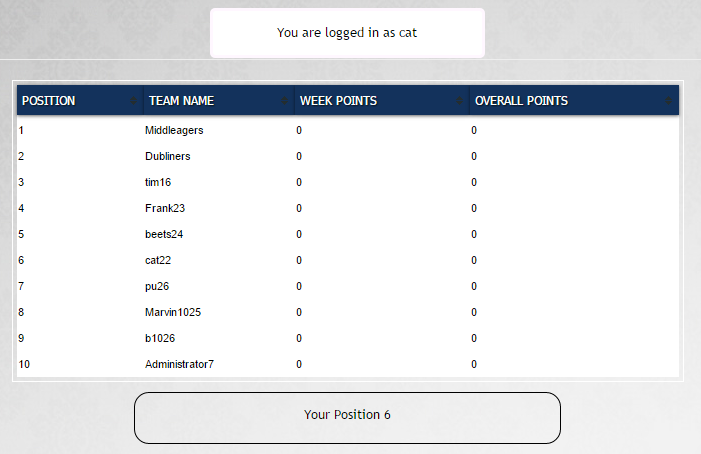
All this information was retrieved using the API and a simple GET request using the appropriate URL. Below is an example of how the full team is retrieved. As can be seen simple local variables were used to decide which the most valuable player was and so on, each iteration the player was checked against the current most expensive player, if he was worth more, he was set as the most expensive. When the request was finished, we then had the most expensive player, top player and so on in variables which were used to update the inner HTML of the DIV elements in the HTML page.





**Viewing the standings**

The user could view his overall position and the top 10 teams in the game in the standings page.

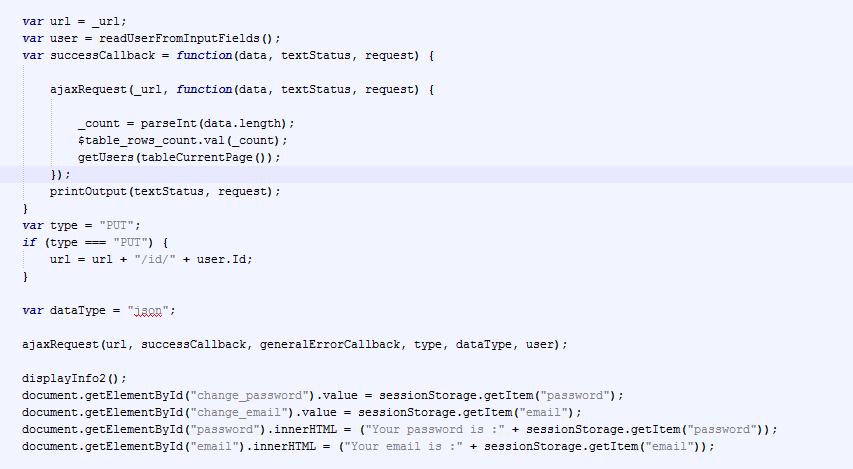


The simple GET request returned the top 10 users based on overall points. A second request was done to retrieve all users, a counter variable was incremented each time a user was found, when the user found matched the current user ID stored in session storage, that counter contained the current users overall position out of all users. Below is an example of the function to retrieve the top 10 users in the database sorted by the highest value for the user’s points.

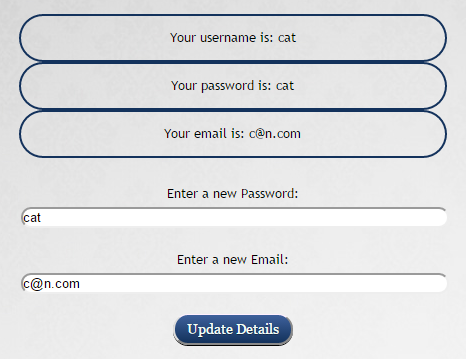


**Changing user details**

A page was created to allow the user change his login details if they wished. The request was made in the usual way, a PUT request was used this time however.

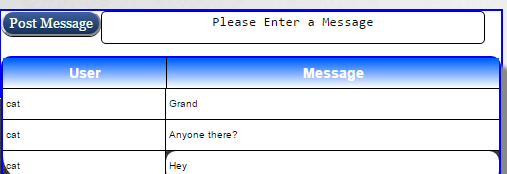


The page was kept simple with HTML DIV elements showing the current users details and text fields allowing the user to enter new details. On click of the button the request created a new JSON array of data, and PUT this data into the user’s details via an AJAX request. Below is an example of the HTML styling in the displayed DIV containing the current details and the fields and button required to change these details.



**Instant messaging in the application**

Social media was integrated into the application in the form of instant messaging on each page. Every functional page on the website has a small neat DIV containing the last 3 posted messages and a text area and button to allow the user post a new message.



The information is retrieved using a GET request in an AJAX function. Only the 3 newest messages are retrieved keeping the feeling of the application being in real time, and keeping the table itself small and efficient. An example of the code can be seen below.



**Making a transfer**

The transfer’s page is the most complex. On initially entering the page a function loads the user’s players. It knows what players to load based off the team ID which was set to session storage earlier. These players are stored in JQuery UI widgets which are resizable and moveable. An example of this can be seen below.



## **Deleting a player from the team**

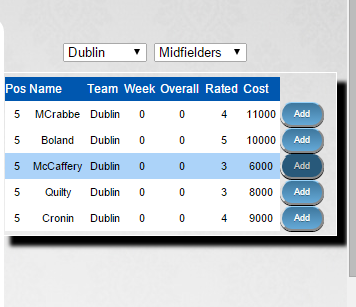
To delete a player, the user clicks on his icon. On click fires a function with the parameter of player position given. We now have the player’s position. We then run a new function iterating through the users team looking for the player who has a position equal to what we passed through originally. When we find it, we take his player ID and pass it to another function. This function removes the player with the ID we pass through from the team with the ID we have stored in session storage! Easy!

We can see below an example of the function to delete a player. It is of course another AJAX request and is of type DELETE. Once the delete function finishes we reload the page after a delay of 600 milliseconds which will reload the entire team again, which of course no longer includes the player we deleted.



**Adding a player to the team**

To add a player to the team, a user simply uses the table on the right to make searches based on team, when he finds a player he wishes to add, he simply clicks the button to add the player.



On click of the add button, a function is fired that retrieves the player ID. Another function is run then to actually add the player. It is of type PUT. If the user has enough budget and the position is vacant the player is added to the team.



## **Dealing with if a user has not enough players in his team**

## One issue that became apparent early on is that if a user creates his team and adds some players, he may not add in the full complement of 8. This should not be allowed. We came up with a simple solution, when the user tries to do anything on the site a check is done, essentially an AJAX request retrieves all the players from the user team and counts them with a simple local variable. If there is not 8 players on the team, then he is redirected to the transfer’s page and he must add into his team additional players.

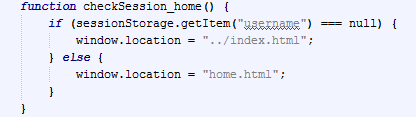
**Creating the user team**

When the user registers, he now has a username and password. At this stage he has no team created in the database. There were a number of solutions to this, but the simplest was to simply do a check when the user tried to view his team or make a transfer if he actually had a team. The user ID is passed into the URL that retrieves all teams, and if the AJAX request retrieves nothing, the user must have no team and one is automatically created for him. It is given a name derived from his username and user ID. We can see below the code to input the new team and the fields it requires in the function below that again.

## 

## **Keeping track of the session**

Another issue was making sure that if the session was ended, that if the user tried to navigate back to that page, he would be brought back to the login page. Also the user should not ever be allowed simply type in for instance the URL for the transfer’s page directly into the browser and be brought there without logging in. A simple way to do this was to use the session storage variables. The first thing that happened on all the pages was a check was done to see if there was a user ID in session storage, if there was not, then the user was re-directed to the login page. This avoided any potential glitches. We can see below a user tries to navigate to the home page, if username is not there then he is brought to the login page again automatically.

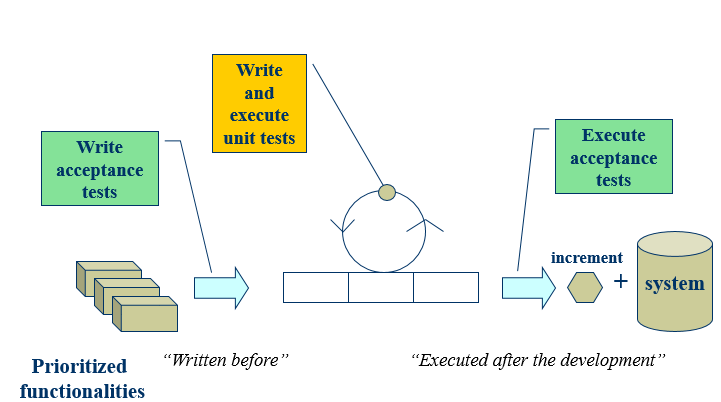


## **Testing**

Testing is the most important part of any software project. Eradicating errors and bugs is vital to the user experience. We came up with some main components to our testing phase.

* Acceptance Testing – this checks if the overall system is functioning as required.
* Unit testing – this is basically testing of a single function, procedure, class.
* Integration testing – this checks that units tested in isolation work properly when put together.
* System testing – here the emphasis is to ensure that the whole system can cope with real data, monitor system performance, test the system’s error handling and recovery routines.
* Regression Testing – this checks that the system preserves its functionality after maintenance and/or evolution tasks.

This testing phase is highly iterative and can be very effective in creating a bug free user experience.

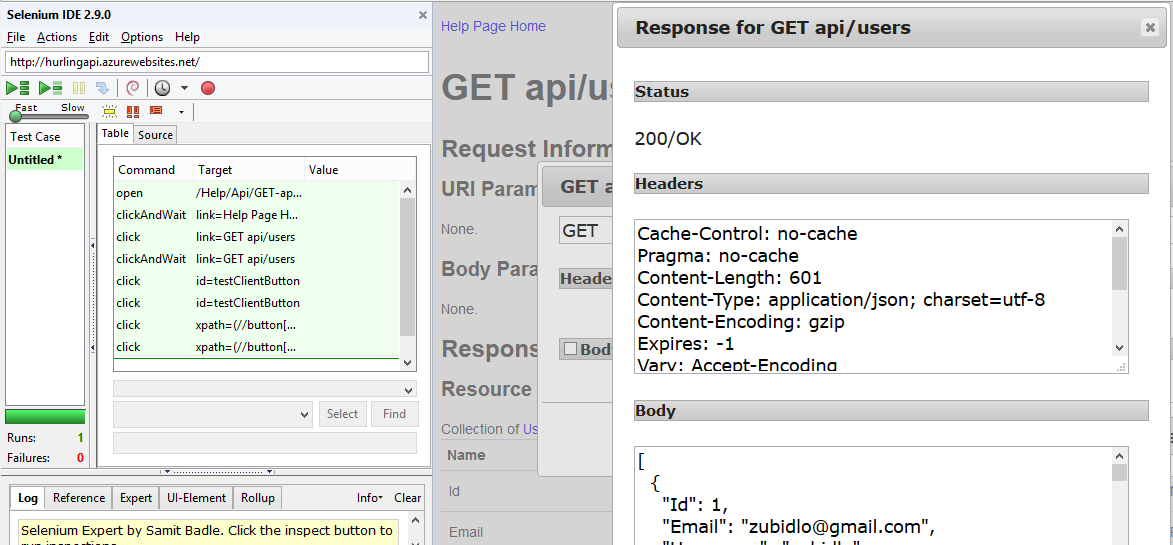
****

Testing the API

We tested the API using Selenium IDE for Firefox. Selenium is an automated web testing tool that allows the user to navigate and interact with a website, and have Selenium track those navigations and changes. You can then choose to let Selenium re-run these while looking for any inconsistencies or errors. The first step was to install the Selenium IDE for Firefox. We then began to test the API help page. This page essentially interacts with the API in the backend and simulates to query and retrieval of the JSON data. Any errors would show up in selenium.

**Test 1**

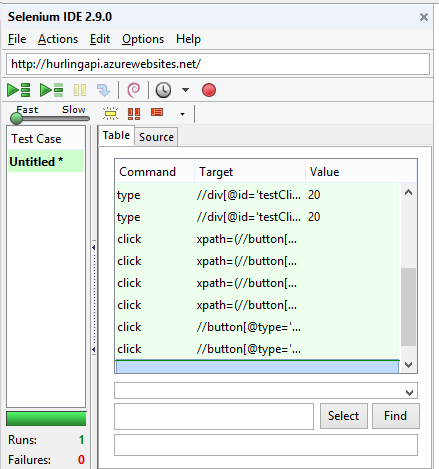
The first test was to try and retrieve all users from the API. This was done by simply setting the URL in Selenium to the main help page URL. Then we navigated to the users and performed the query as normal. The query was then re-run in Selenium.



We can see that the test passed successfully. This means the navigation and retrieval of the JSON data from the API was error free.

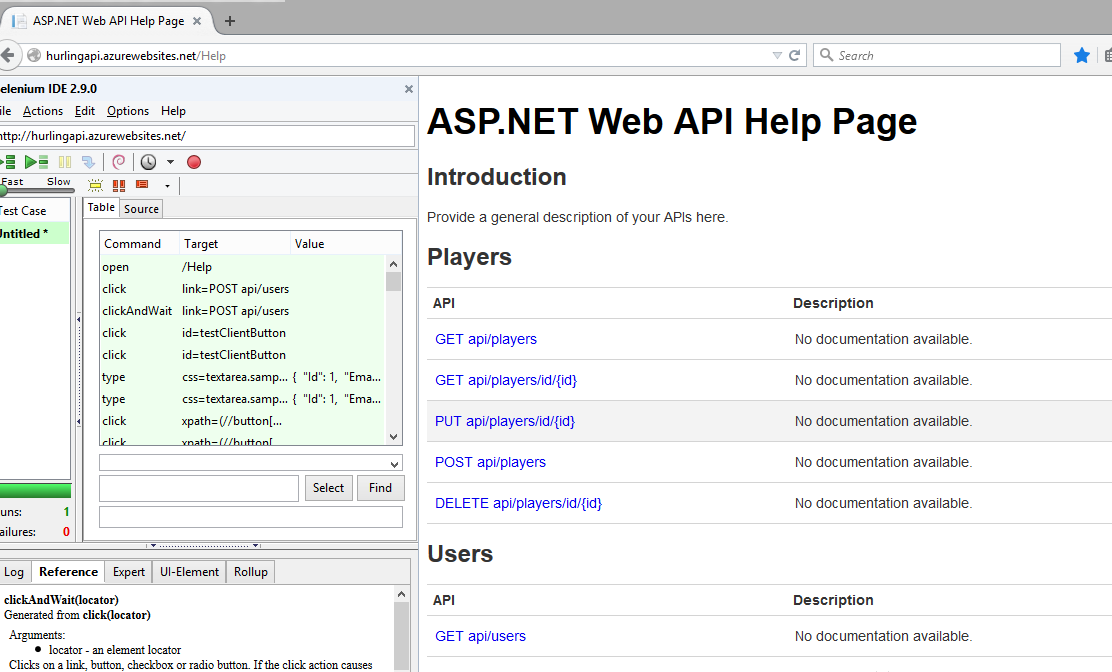
**Test 2**

The second test was to input a player into a specified user’s team via the API. We simulated the deleting of player with ID 20 from the team with ID 2. We then deleted the player so that when Selenium ran, it would not get an error trying to re-add a player that is already in a team! As we can see the test was successful and we received no errors.



**Test 3**

The final test we ran on the API was a little more complex. It involved adding a new user, creating a team for him, then adding a player to that team. We created a new user, then a new team, and added player with ID 1 to that team, who’s ID was 1061. The user ID was 1030. The test ran successfully and we encountered no errors.



**Testing the application Website**

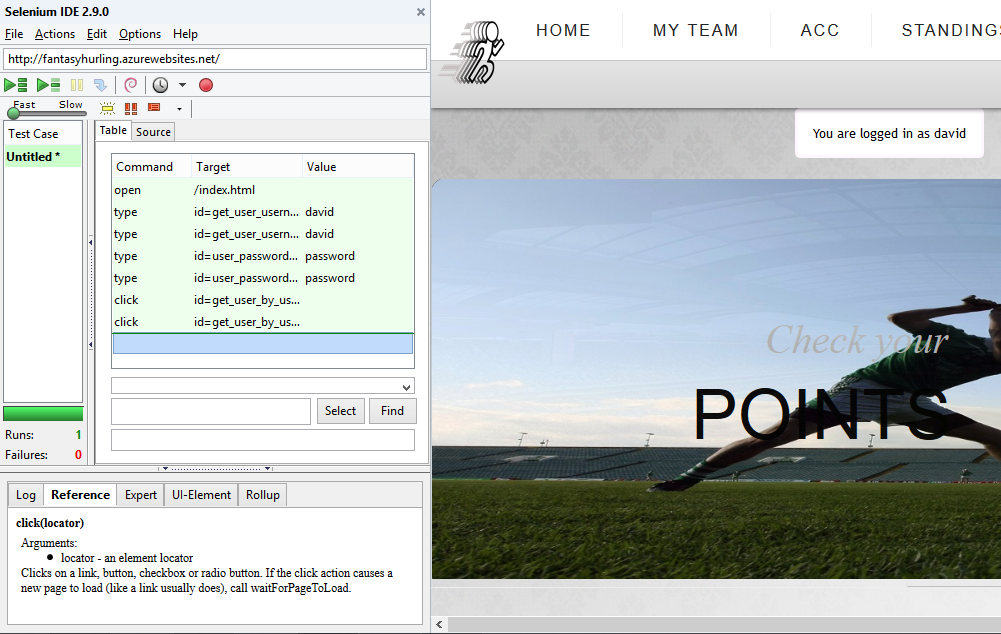
In regards the testing of the site of the site itself, there were a few factors deemed as very important and were considered to be so vital as even a small failure was unacceptable. These were as follows:

* The login of the user should be flawless
* Registration should be flawless
* Changing user details
* Deleting a player
* Adding a player

Due to the high level of importance for all of these interactions with the site, we decided to perform Selenium tests on all 5 scenarios.

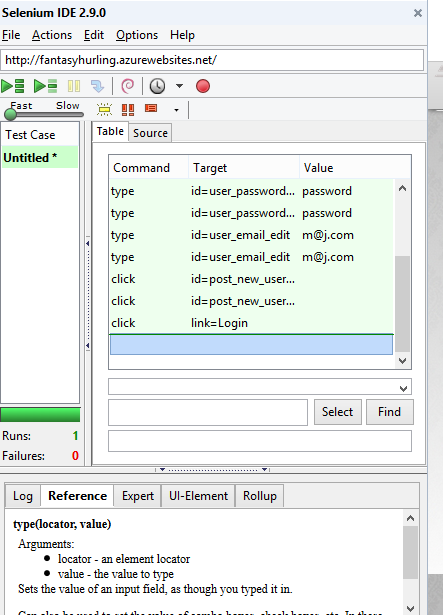
**Test 1**

The first test was to see if the user could login properly and without an errors. We ran the Selenium test and we can see from the results below the test was successful with no errors:



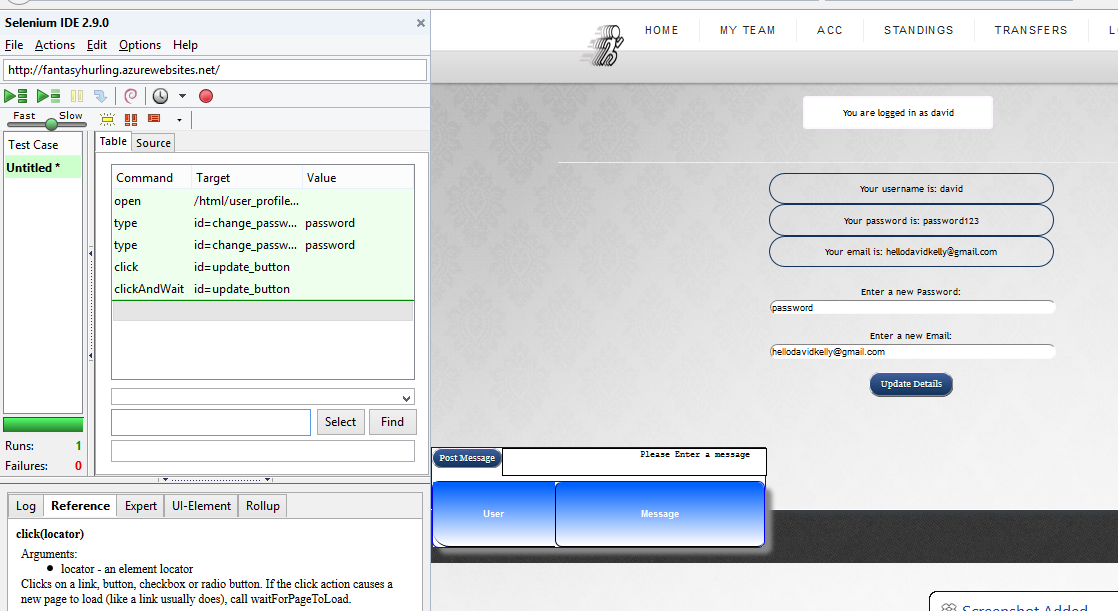
**Test 2**

The second test we ran was to make sure the user could register with no issues. Again we ran the selenium test and as expected there were no problems that needed to be resolved. We can see the output of the successful test below:



**Test 3**

The next test was to see if the user could change their details without issue. We ran the Selenium test and the results were again as expected, the test was successful with no errors. We can see the output below.



**Test 4**

We next ran a test to see if the user could delete a player from his team. The results were expected to be successful with no failures and this was indeed the case.



**Test 5**

The last test was to see if we could add a player into the team, the results were as expected and with no failures. We removed player with position 2 and the re-added him capturing the input with Selenium. We then stopped the capture, removed him and used Selenium to try automate his addition to the team again. It worked with no issues.



## **Conclusion**

We did not choose a trivial project. Being unexperienced developers we didn’t foresee the complexity and challenges involved in developing a fantasy sport web application. After realizing the scope of the project we changed our initial ideas how to approach this project development. We decided to follow prototyping methodology implementing and test one functionality at the time. Hoping to learn from each iteration and deliver a modern optimized rich user experience application.

In this occasion we decided to use the most modern technologies and designs. We are building a Web API Service so that different clients could get access to CRUD methods. This way the additional clients can get developed easily. For example in the future we can easily implement Android client or Windows 8 metro client for our fantasy hurling application. We are using Entity Framework ***[14]*** to automatically map database table rows to entity objects so we don’t need to write one sql query. We are implementing our Web API using .NET Web API 2 framework which is powerful toolkit capable of asynchronous responses, cross-origin requests and much more. We are deploying the application up on Microsoft Azure Cloud.

This project is an opportunity for us to learn new technologies and get valuable firsthand experience and skills in web development.

## **References**

1. What is Prototype model- advantages, disadvantages and when to use it? 2014. What is Prototype model- advantages, disadvantages and when to use it? [ONLINE] Available at: <http://istqbexamcertification.com/what-is-prototype-model-advantages-disadvantages-and-when-to-use-it/>. [Accessed 14 December 2014].
2. HTML5. 2014. HTML5. [ONLINE] Available at: <http://www.w3.org/TR/html5/>. [Accessed 14 December 2014].
3. Introduction to CSS3. 2014. Introduction to CSS3. [ONLINE] Available at:<http://www.w3.org/TR/2001/WD-css3-roadmap-20010523/>. [Accessed 14 December 2014].
4. Ajax | MDN. 2014. Ajax | MDN. [ONLINE] Available at: http. [Accessed 14 December 2014].
5. JQuery. 2014. JQuery. [ONLINE] Available at: <http://jquery.com/>. [Accessed 14 December 2014].
6. Menu | JQuery UI. 2014. Menu | JQuery UI. [ONLINE] Available at:<http://jqueryui.com/m/>. [Accessed 14 December 2014].
7. Sublime Text: The text editor you'll fall in love with. 2014. Sublime Text: The text editor you'll fall in love with. [ONLINE] Available at:<http://www.sublimetext.com/>. [Accessed 14 December 2014].
8. Aptana. 2014. Aptana. [ONLINE] Available at: <http://www.aptana.com/>. [Accessed 14 December 2014].
9. Storage Documentation | Azure. 2014. Storage Documentation | Azure. [ONLINE] Available at: <http://azure.microsoft.com/en-us/documentation/services/storage/>. [Accessed 14 December 2014].
10. Azure Regions. 2014. Azure Regions. [ONLINE] Available at:<http://azure.microsoft.com/en-us/regions/>. [Accessed 14 December 2014].
11. Web API - Wikipedia, the free encyclopedia. 2014. Web API - Wikipedia, the free encyclopedia. [ONLINE] Available at:<http://en.wikipedia.org/wiki/Web_API>. [Accessed 14 December 2014].
12. node.js. 2014. node.js. [ONLINE] Available at: <http://nodejs.org/>. [Accessed 14 December 2014].
13. ASP.NET Web API | the ASP.NET Site. 2014. ASP.NET Web API | the ASP.NET Site. [ONLINE] Available at: <http://www.asp.net/web-api>. [Accessed 14 December 2014].
14. Cross-origin resource sharing - Wikipedia, the free encyclopedia. [ONLINE] Available at: <http://en.wikipedia.org/wiki/Cross-origin_resource_sharing>. [Accessed 14 December 2014].
15. Entity Framework. 2014. Entity Framework. [ONLINE] Available at:<http://msdn.microsoft.com/en-ie/data/ef.aspx>. [Accessed 14 December 2014].

Podila. (2013). *HTTP: The Protocol Every Web Developer Must Know.* Available: http://code.tutsplus.com/tutorials/http-the-protocol-every-web-developer-must-know-part-1--net-31177. Last accessed 2nd April 2015

HTTP/1.1. (2015) *Hypertext Transfer Protocol -- HTTP/1.1*. [ONLINE] Available at:<http://www.w3.org/Protocols/rfc2616/rfc2616>. [Accessed 04 April 2015].

URI (2015). Uniform Resource Identifiers. [ONLINE] Available at: <https://www.ietf.org/rfc/rfc2396.txt>. [Accessed 05 April 2015].

HTTP Tutorial. (2015). HTTP Tutorial. [ONLINE] Available at:<http://www.tutorialspoint.com/http>. [Accessed 05 April 2015].