

**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: Group Project Prototype Doc**

**Due Date: 19/12/14**

**Date Submitted: 10/12/14**

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**

[**Project Description** 1](#_Toc406345137)

[**What We Will Look at in this document** 2](#_Toc406345138)

[**Development Methodology** 3](#_Toc406345139)

[**Project Plan** 5](#_Toc406345140)

[**Use Cases** 6](#_Toc406345141)

[User checks current score 6](#_Toc406345142)

[User checks fixture 7](#_Toc406345143)

[User uses social media aspect 8](#_Toc406345144)

[User creates team 9](#_Toc406345145)

[User makes transfer 10](#_Toc406345146)

[**Sequence Diagrams for Use Cases** 11](#_Toc406345147)

[User checks score 11](#_Toc406345148)

[User checks fixtures 11](#_Toc406345149)

[User posts message 12](#_Toc406345150)

[User chooses team 12](#_Toc406345151)

[User makes a transfer 13](#_Toc406345152)

[**Wire framing** 14](#_Toc406345153)

[Login Page 14](#_Toc406345154)

[Registration Page 15](#_Toc406345155)

[Team statistics page 16](#_Toc406345156)

[Player transfer page 17](#_Toc406345157)

[Standings Page 18](#_Toc406345158)

[Social Media area 19](#_Toc406345159)

[**Work Flow Plan** 20](#_Toc406345160)

[**What has been done?** 21](#_Toc406345161)

[**What is still to do?** 22](#_Toc406345162)

[**DESIGN and TECHNOLOGY details** 23](#_Toc406345163)

[**Conclusion** 26](#_Toc406345164)

[**References** 27](#_Toc406345165)

**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 15 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:**

## **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]***.

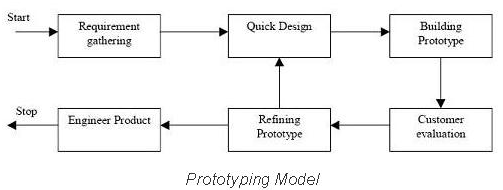


Figure 1 Prototyping SDLC work flow

**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)

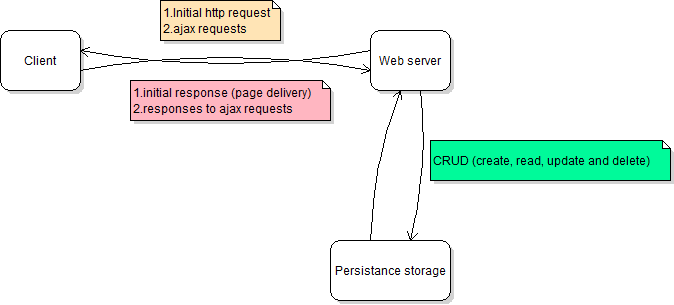


Figure 2 Web Application general design

## **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

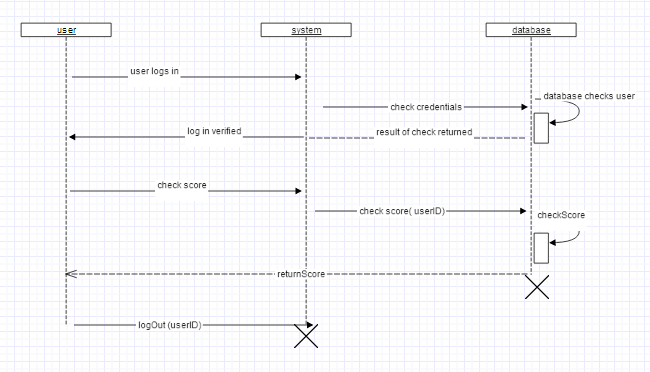
2a: User has not enough funds to make transfer

2b: User selects and invalid team choice

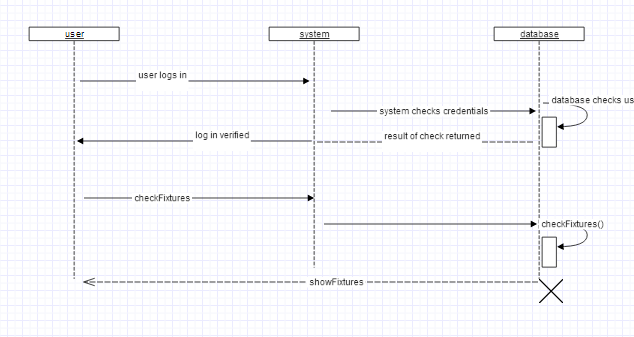


## **Sequence Diagrams for Use Cases**

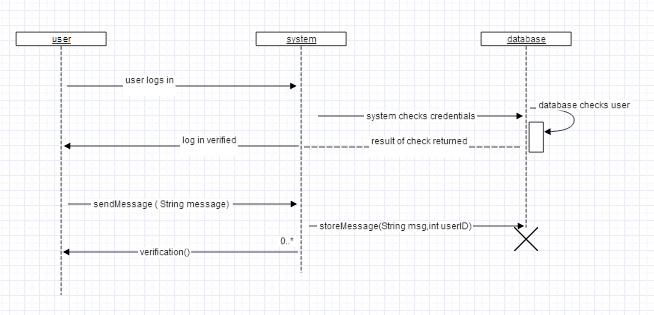
### User checks score



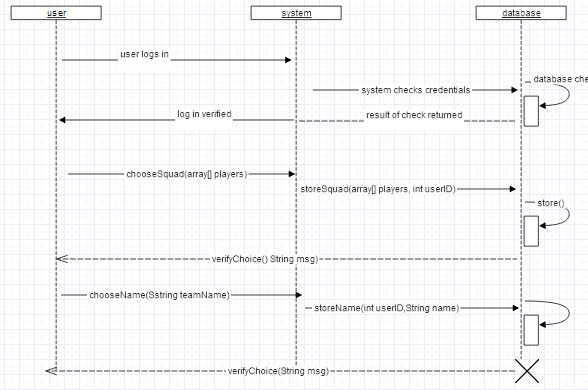
### User checks fixtures



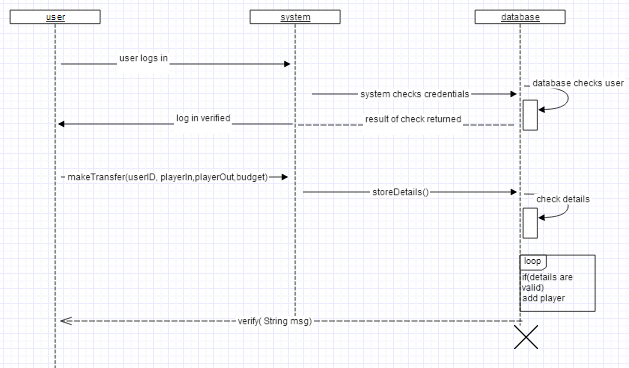
### User posts message



### User chooses team

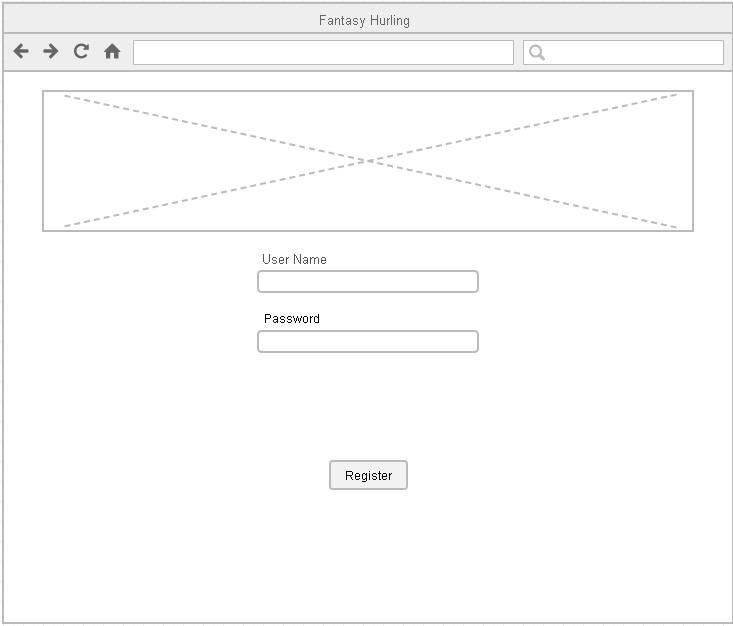


### User makes a transfer



## **Wire framing**

### Login Page

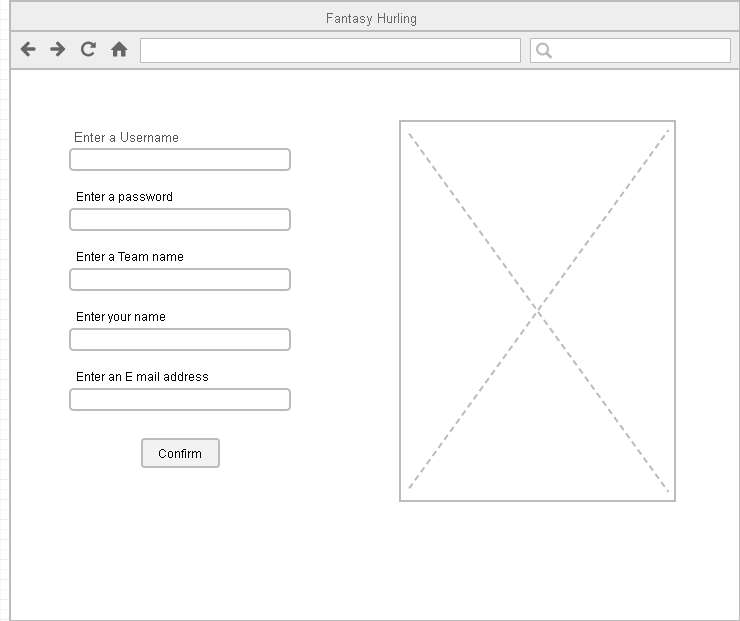


On this page the user can log into the system. They enter a username and password and are logged in. There is an image also.

Elements needed:

* 2 Text boxes
* One button
* One image

### Registration Page

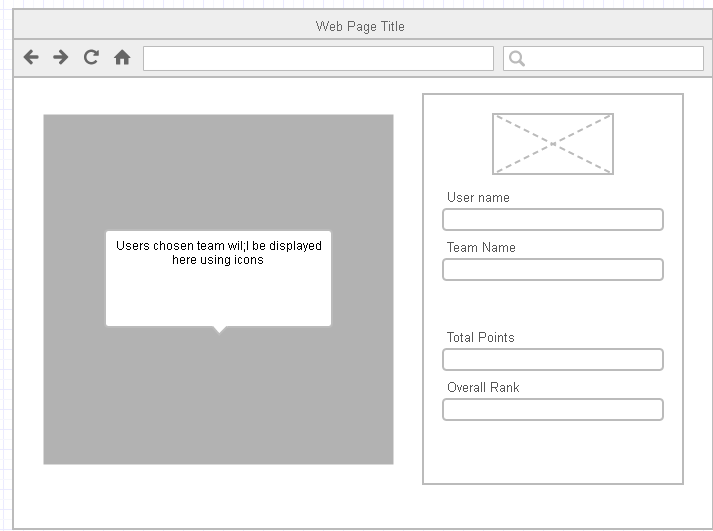


On this page the user can register if they have not already.

Elements needed:

* 5 Text boxes
* One button
* One image

### Team statistics page

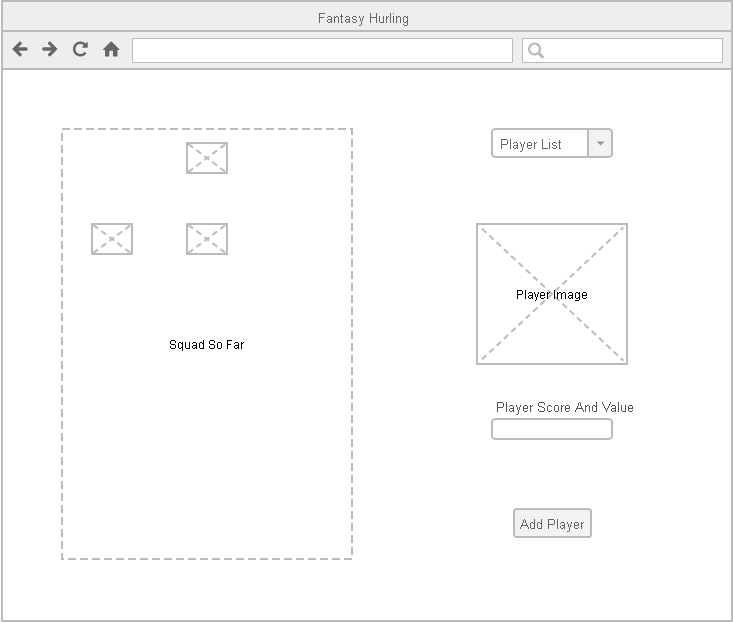


On this page the user can view their team’s current standing and overall points. It shows the user name and team name and the team the user has selected on the left

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

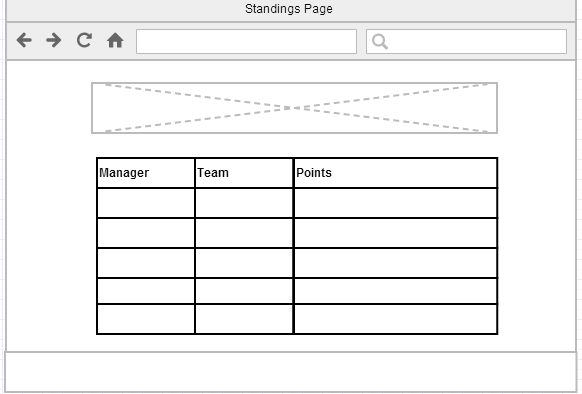


On this page the user can make transfers if they want to. They can search through the database of players, remove a player from their team, and see player score and value. Nothing is confirmed till the user clicks the add player button.

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

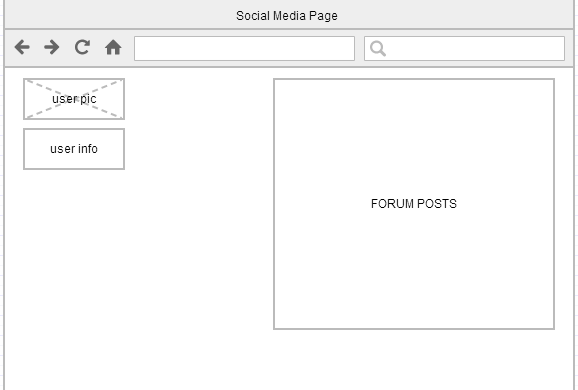


On this page the player can view the top players in the game by points scored. He may be in this list, it will be a simple SQL query displayed in a HTML table.

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

## **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

## **What has been done?**

* Web page prototype with working use case of AJAX CRUD request to Web API. Web page is maintained on local hosts only.
* Web API prototype capable of answering cross-origin requests ***[14]*** and generating SQL queries for requested CRUD operations. Web API prototype is deployed on Azure: <http://fantasyhurling.azurewebsites.net/>  
  API is in very early stage of development and it’s not optimized.
* Database prototype deployed on Azure Storage server
* This documentation.
* Project cooperators diaries.

**Assess Project Worth (Level 1 task)**

* Task 1: Create list of project necessities (Level 2 Task) (COMPLETE)
* Task 2 : Check internet for technologies needed (Level 3 Task ) (COMPLETE)
* Task 3 :Check copyright on technologies and official logos (COMPLETE)
* Task 4 :Check price on project necessities (COMPLETE)
* Task 5 :Decide on project worth doing or not (COMPLETE)
* Task 6 :Choose a project supervision (COMPLETE)

**Design phase**

* Task 7: Research design method and select one ( Level 1 task ) (COMPLETE)
* Task 8 :Create specifications in detail ( Level 2 Task ) (COMPLETE)
* Task 9 :Plan staff workload and duties ( Level 3 Task ) (COMPLETE)
* Task 10:Plan overall timescale (COMPLETE)
* Task 11:Choose technologies to suit project (COMPLETE)
* Task 12:Decide on web site visual design and logo (COMPLETE)
* Task 13:Determine if budget is enough (COMPLETE)

**Implementation Phase**

* Task 14:Write the code for the project ( Level 1 Task ) (COMPLETE)

## **What is still to do?**

The main body of the coding still needs to be completed. We need to link our site to the Java backend and the SQL database, but we are on schedule comfortably.

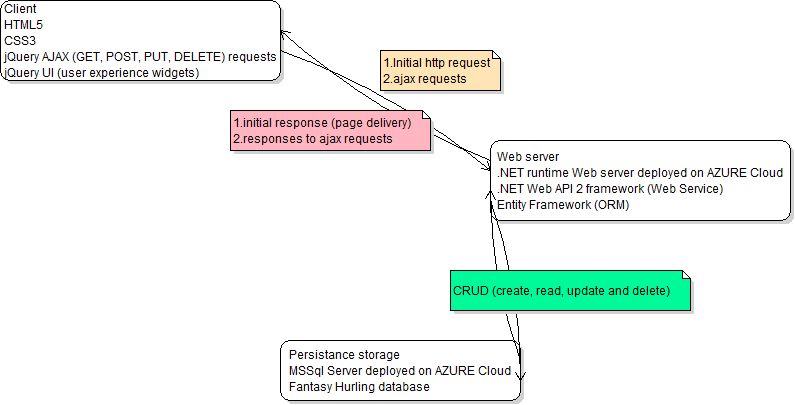
* Task 15:Design the web site with HTML and CSS and JSON ( Level 2 Task ) (January 17th - March 15th)
* Task 16:Logic written in C# (January 17th)
* Task 17:Tie the two together (March 17th – March 25th)

Testing phase

* Task 18:Select a method of testing ( Level 1 Task ) (March 30th)
* Task 19:Check internet for suitable documentation to back up testing method ( Level 2 Task )(March 30th)
* Task 19:Test project using JUNIT and other testing methods ( Level 3 Task )(April 3rd-April 30th)
* Task 20 :Roll out project (May 1st 2015)

Keep documentation up to date including diary (Level 3 Task) (Ongoing September 2014-April 2015)

## **DESIGN and TECHNOLOGY details**



*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.

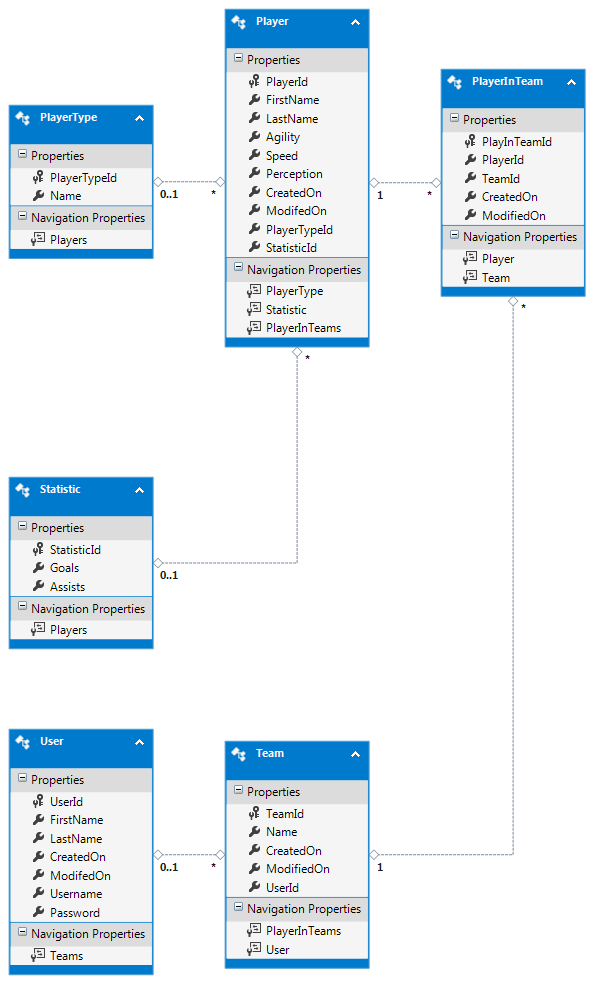
**Front-End**

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.

## **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].