Digital Media Technologies

Assessment Task 4

Research Project

Name: Stuart Bryson
Student #: 98082365

Table of Contents

Research Project Details

Motivation

For Assignment 4 of the Digital Media Technologies subject at UTS, we were to create a project that "represents the realisation of your Technology Learning Contract and MUST demonstrate your newly acquired skills and knowledge of your chosen technology(s) of interest". This document details the project that I developed.

Document Structure

This document contains 3 sections. The first section details the original project aim, outlined in both the Learning Contract and System Plan, and how the final Research Project has deviated. The second section is a User Manual that provides detailed instructions illustrating how to use the site. The last section is a technical discussion that describes some of the technical design decisions of the site. It provides insight into how the site works, and specifically how the technologies chosen were integrated into the site.

URI

The project is available on the Internet at this location: http://bryson.heironym.us/Hockey/home.php

Conformance

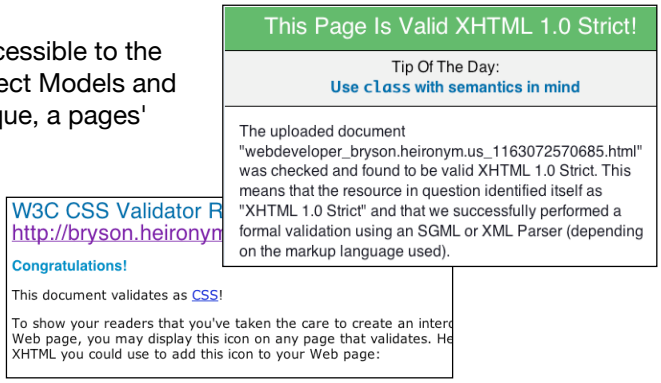
The entire website conforms to both XHTML strict 1.0 and CSS2. The online w3c validators were used to verify that every single page accessible on the website was validated. This validation took into account both PHP generated XHTML and modifications to the Document Object Model.

PHP Generated XHTML Validation

It was important when performing these validations to remember that a PHP file can generate countless variations of a web page. Specifically in this project, the XHTML will differ greatly depending on whether or not the administrator is logged in. For this reason, every page was tested with the administrator being logged out and logged in. As the login procedure means that the XHTML will not be available as a live page to the validator, the 'local HTML' was sent to the validator for verification.

Modified DOM Validation

In the same way the logged in PHP pages were not accessible to the validator, any pages that have modified Document Object Models and or CSS must also be validated. Using the same technique, a pages' 'local HTML' was sent to the validator for verification.



Project Goals and Deviations

Learning Contract

This research is a practical realisation of the goals that I had at the start of the subject. That was to use the latest web technologies to build an Ice Hockey website for the league I am involved with. The Learning Contract (Bryson, 2006a, p.4) summarised the original goal:

> *The project I wish to embark on will be a website for an ice hockey league I am involved with. The website will be a database driven site with access to player profiles, venue information, the draw and game results.*
>
> *The site will also have an administration component that will provide an administrative user with the ability to enter in player information and game results. This information will be inserted into the database and provide the website with various reports and statistics on the players, teams and games.*

In order to achieve these goals, I was required to use various technologies that I had little or no experience with. In the Learning Contract, I detailed 5 technologies that the Research Project would be based on. These were: XHTML, CSS, JavaScript, PHP and MySQL. The specifics of my experience with these technologies can be found in the Learning Contract.

Valid XHTML and CSS are used on every page of the project for content structure and layout respectively. MySQL is used to store all dynamic content such as game details and player statistics. This data is retrieved by PHP that then generates XHTML. While the every day user of the site does not require any JavaScript to be running, the administrator relies heavily on it for modification of the data.

The Learning Contract also detailed some other technical considerations and methodologies that I hoped to integrate into the project. These included a security model, AJAX enhancements and browser compatibility. The project implements a simple security model for the administrator login. It also makes use of AJAX for enhancing the administration experience. Lastly the everyday user can view the site using Firefox, Safari and the latest version of Internet Explorer. The administration component has been tested in Firefox and Safari. I doubt, however, that the administration component works in Internet Explorer. This is consistent with the Learning Contract which states that I did not want to "sacrifice functionality of my website in order to support archaic browsers – particularly in the administration component" (Bryson, 2006a, p. 6)

This Research Project has achieved all the Goals set out in the Learning Contract. The technologies described and extra features, while extremely ambitious, have all been realised in this Research Project.

System Plan

In the System Plan, I outlined the System Architecture that would be employed for the project (Bryson, 2006b, p.2). This architecture was consistent with the Learning Contract and has been used to implement the Research Project. The System Plan also outlined the basic insert, update and delete paradigm for the entities in the system and the use of one PHP file for each logical page. All the entity pages in the system have been implemented in this manner.

While the Research Project realises all of the goals in the Learning Contract, there are various features of the System Plan that have not been reached. They are as follows:

Along with the insert/update/delete paradigm, the Plan also outlined the desire to have detail pages for each entity. One example is the player statistics page. These pages have not been implemented for 2 reasons. Firstly, some of the data that was to be presented to the user on this details page has actually been implemented on the main summary page. For example, player statistics such as the number of goals and assists that a player has earned across the season is presented on the main players page. Secondly, there was simply not enough time to implement all of these pages. I decided it was better for me to focus on the pages that I had, get them working well and implement the other pages at a later date.

Secondly, some of the navigational components of the site were not implemented. Once again, this was simply due to a lack of time and will be implemented at a later date.

All of the features that were marked as 'extra' in the System Plan have also not been realised, however these were never intended to be implemented for this version but rather were looking ahead to further development of the site.

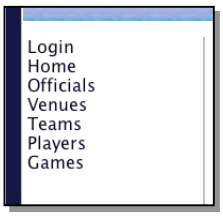
This Research Project has achieved almost all of the functionality laid out in the System Plan. While the System Plan, like the Learning Contract, was ambitious, I have been able to implement the majority of the functionality that was proposed.

User Manual

This user manual will provide the reader with an understanding on how to use the East Coast Super-league web site. This includes both general use of the site such as browsing the site for game details or player statistics and also administrative features for entering and updating league information.

General Use

The navigation for the site is accomplished through the use of the navigation side bar. This side bar provides quick access to all the main pages of the site. To access a page, simply click on one of the site bar options. The functionality of the pages is as follows.



Login

The Login page is used only for administrators. See the Administration section.

Home

The home page provides important introductory details to the users of the site, including international and local users who may be interested in the league. This home page may be updated from time to time to reflect any important changes in the league or convey important messages to the site users.

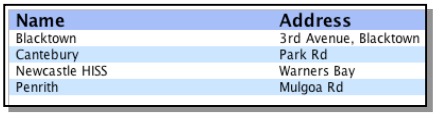
Name
Les Adams
Michael Duncan
Hugh Ferrar
Warrick Griffith
Josh Herbert
Sam Herbert
Richard Manco
Andrew McMahon
Steve Morgan
Jason Scott
Jose Zvonicek

Officials

The officials' page provides a list of all officials currently registered with the league. The list can be sorted by last name by clicking on the Name column heading. To sort in the opposite direction, click on the heading again.

Venues

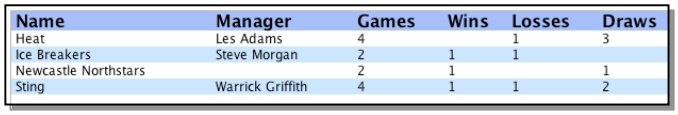
The venues page provides a list of all venues registered with the league. The list can be sorted by Name or Address by clicking on the respective column heading. To sort in the opposite direction, click on the heading again.



Name	Address
Blacktown	3rd Avenue, Blacktown
Cantebury	Park Rd
Newcastle HISS	Warners Bay
Penrith	Mulgoa Rd

Teams

The teams' page provides a list of all the teams currently registered with the league. A team can also have an optional manager who is a member of the officials list.

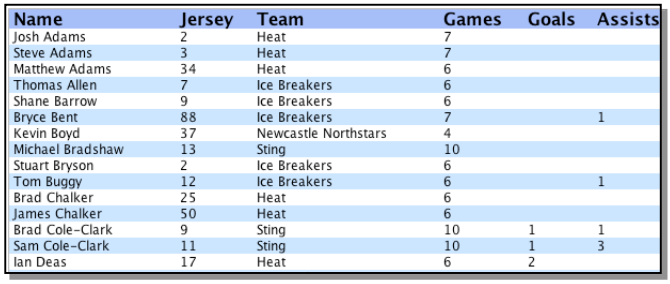


Name	Manager	Games	Wins	Losses	Draws
Heat	Les Adams	4		1	3
Ice Breakers	Steve Morgan	2	1	1	
Newcastle Northstars		2	1		1
Sting	Warrick Griffith	4	1	1	2

As well as listing the teams, this page provides some high level statistics about the teams. It displays how many games each team has played, and how many of those games each team has won, lost or drawn. The list can be sorted by any of the columns by clicking on a column heading. Clicking on the column heading again will reverse the sort direction.

Players

The players' page provides a list of all the players currently registered with the league. A player can also have an optional Team association and also an optional jersey number. If these are not provided, a player can still play. Indeed, a player from one team can still play for another and with a different jersey number than the one displayed here. The team association and jersey number are simply to describe the players' normal playing behaviour.



Name	Jersey	Team	Games	Goals	Assists
Josh Adams	2	Heat	7		
Steve Adams	3	Heat	7		
Matthew Adams	34	Heat	6		
Thomas Allen	7	Ice Breakers	6		
Shane Barrow	9	Ice Breakers	6		
Bryce Bent	88	Ice Breakers	7		1
Kevin Boyd	37	Newcastle Northstars	4		
Michael Bradshaw	13	Sting	10		
Stuart Bryson	2	Ice Breakers	6		
Tom Buggy	12	Ice Breakers	6		1
Brad Chalker	25	Heat	6		
James Chalker	50	Heat	6		
Brad Cole-Clark	9	Sting	10	1	1
Sam Cole-Clark	11	Sting	10	1	3
Ian Deas	17	Heat	6	2	

As well as listing the players, this page provides some high level statistics about the players. It displays how many games each player has played, and also a total number of goals and assists the player has achieved. The list can be sorted by any of the columns by clicking on a column heading. Clicking on the column heading again will reverse the sort direction.

Games

The games page provides a list of all the games currently registered with the league. Each game listed here will provide information about the game including the date, venue and teams involved. If the game's results have been entered into the system, there will be a link to view the results for that game and the final score for the game should also be displayed. Clicking on the link to view the results will take you to the game results page.

The list can be sorted by any of the columns by clicking on a column heading. Clicking on the column heading again will reverse the sort direction.


Ice Breakers: 3 Sting: 2

Game Details

Date	17 Apr 06
Venue	15
Game #	4
Start Time	19:45
End Time	20:05
Period 1	20
Period 2	20
Period 3	25
Referee	Richard Manco
Linesman	Jose Zvonicek
Linesman	
Announcer	
Timekeeper	Warrick Griffith
Scorekeeper	Steve Morgan
Goal Judge	
Goal Judge	
Home Official	Steve Morgan
Away Official	Warrick Griffith

Home Players

Team	Name	Jersey
Ice Breakers	Dave Furguson	1
Ice Breakers	Stuart Bryson	2
Ice Breakers	Ben Denyer	5
Ice Breakers	Shane Barrow	9
Ice Breakers	Michael Schlamp	11
Ice Breakers	Thomas Buggy	12
Ice Breakers	Luke McDermott	17

Game Details

This page is only accessible from the main Games page. If a games result has been entered into the system, a link from the Games page will provide you with all the results of that game.

The game details page will detail the overall results of the game, where it was held, who refereed and other general details. It also lists all the specific details including which players played for each team, the shots on goals statistics, penalties and goals.

Administration

The administration component allows you to enter, update and delete league information. It provides administration of officials, venues, players, teams and games.

System Requirements

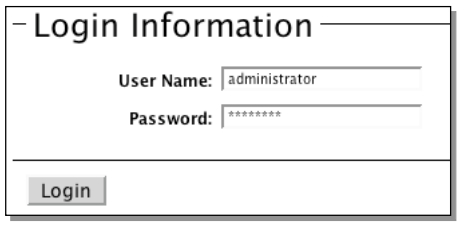
While for general use the website works on almost all browsers, the administration component has only been tested in Firefox v2.0. It is possible that it also works in Internet Explorer 7, Safari and others however it is not recommended. As Firefox is free and available on all platforms, it is highly recommended that you download and install it. It is available at http://www.mozilla.com

Login

Before anything can be administered, you must log in with a valid administration user name and password. Your log in details should have been provided to you when the site was set up. Write them here for further reference.

Username: *tim*
Password: *password*

To login, click on the login option in the side bar. Enter in your username and password and click the Login button.



If your login details are correct, you will see a message telling you that you are logged in and the Login navigation option will change to Logout. If your details are incorrect, you will see a message telling you that you entered an invalid username or password.

NOTE: Tim - for testing purposes only, there are also a few administration buttons available on the login page after you have logged in. These buttons are not intended for release.

General Workflow

Once you are logged in, you will be able to administer the rest of the site. There is a general workflow for most of the site that allows you to add, edit, and remove details from the league as required. When logged in, each page will look similar as when you are not logged in. However, there will be some subtle differences such as the addition of three different icons. The functions of these icons are:

Add. This icon gives you the ability to add a new record. It should appear above the list of registered items. Clicking this icon will display a form for entering in the new record below the list of registered items. Each item in the form will be either Required or Optional. The Required data will be in bold type.

Delete. This icon gives you the ability to delete an existing record. There will be a delete icon for every record and it will appear at the start of the row. Clicking this icon will immediately delete the corresponding record. *NOTE: The site does NOT ask for confirmation of delete.*

Edit. This icon gives you the ability to edit an existing record. There will be an edit icon for every record and it will appear at the start of the row. Clicking this icon will display a form for editing the records details. The form should be filled with the records current data. DO NOT use the edit option as a workflow for logically deleting one record and adding another.

If the add or edit button are clicked one after the other, you will notice the form for adding and editing will be updated. *NOTE: Any data you have entered into the form, either an add or an edit, will be lost if you click the add or edit icons again before that data has been submitted to the database.*

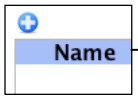
If at any stage you enter in erroneous data, the input element will indicate this by drawing a red box around the element. By mousing over the element, the tooltip will explain what is in error. All erroneous data must be corrected before the form is submitted.

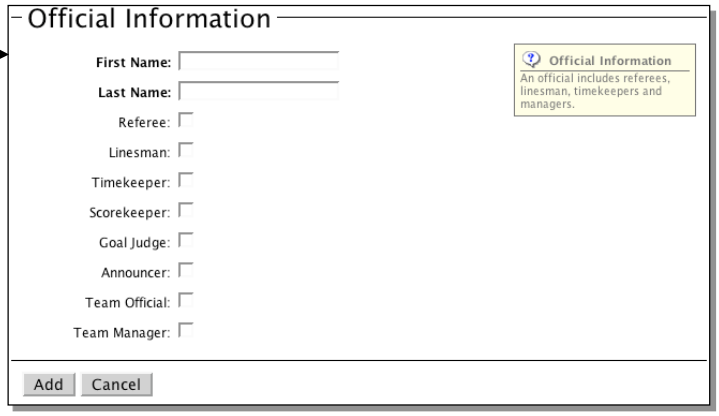
Page Details

Following is a detailed list of the administration options for each page.

Officials

To add a new official, click on the add icon. This will display the add form under the officials list.





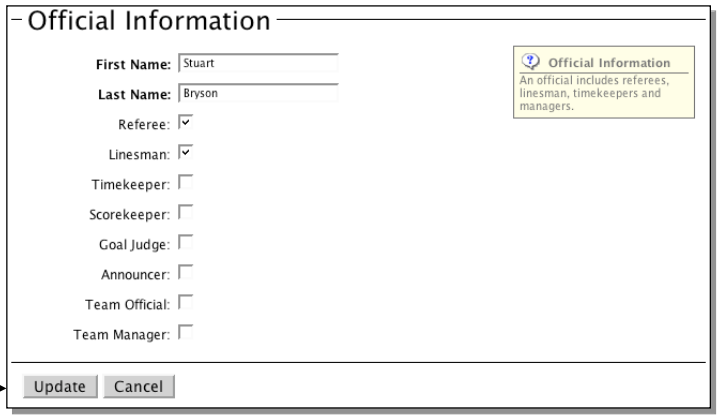
Enter in the required details of first and last name. Then tick any functions that this official is capable of. For example, this official may be a Referee and a Linesman but not a Team Manager.

The difference between a team official and a team manager is that a manager can be registered with a particular team when adding and editing team information. An official and a manager can both be registered on a game sheet as being the team official for that game.

Once you have filled out the form, click Add. This will submit the record to the database and it should now appear in the list of registered officials.

To edit an official, click on the edit icon for that official. Click cancel if you wish to leave the record unchanged.

Finally, to delete an official, click the delete icon.



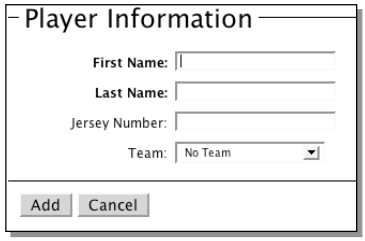
Venues

To add a new venue, click on the add icon. This will display the add form under the venues list.

Enter in the required Venue Name and the optional Address. Click the Add button and the venue should be added to the list.

Click on edit or delete to update and remove the record from the list of registered Venues.

Players



To add a new player, click on the add icon. This will display the add form under the players list.

Enter in the required First and Last names of the player. You can also enter in an optional Jersey Number and Team. Both the Jersey Number and the Team are optional default values and can be changed on a per game basis. These extra default values will facilitate you when entering in the results of a game but are not strictly necessary.

Click on edit or delete to update and remove the record from the list of registered Players.

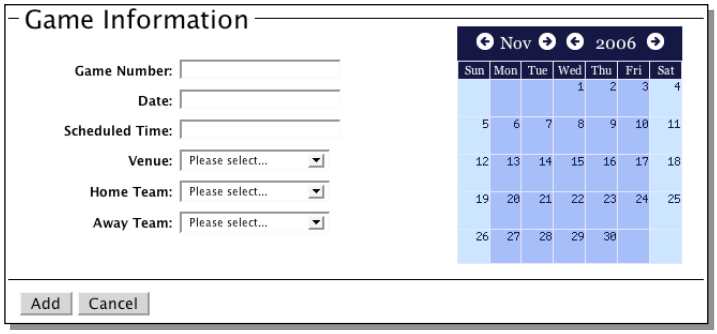
Teams

To add a new team, click on the add icon. This will display the add form under the teams list.

Enter in the required team name and select an optional manager. The manager list is determined by which registered officials have been marked with the Team Manager ability.

Click on edit or delete to update and remove the record from the list of registered Teams.

Games



To add a new game, click on the add icon. This will display the add form under the games list.

All of the information in this form is required. Enter in the Game Number, Date, Time, Venue and Teams.

The Game Number must be a unique number across the league. As you enter in a number, the form will validate that number and ensure that it is not already being used by another game. If it is, it will be marked as an error and you must replace it with a unique number.

The Date must be entered in the form of YYYY-MM-DD. The reason for this is to avoid any international confusion between the Australian and US date format. Regardless, there is a calendar to the right of the form that allows you to browse for a date. By clicking on a particular date in the calendar, the date in the form will be updated to reflect your choice.

The venue list is retrieved from the list of venues and similarly so are the home and away team lists. If either of these lists is incomplete, you must return to the appropriate page and enter their information before proceeding with entering in a game.

Click on edit or delete to update and remove the record from the list of registered Games. For every registered game, there should also be a link to enter results for the game. Once a games results have been entered, you can no longer edit or delete that games details. Further, the edit and delete icons will not appear for that game.

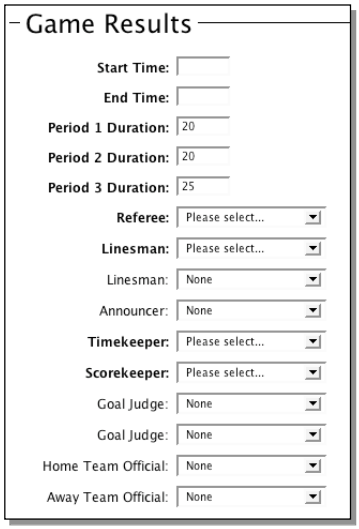
Game Details

To enter a games details, click on the enter results link next to the particular game. This will take you to a new page where you can enter in a game's details. You cannot delete a game or update a game's details once you have entered them in. This data is considered historical data and we do not allow editing of it.

The game details page is the most complex page for data entry. The page is divided into 3 clear sections; General Game Details, Home Team Details and Away Team Details. The Home and Away Team Details work in exactly the same way as each other but they are dealing with a different team.

General Results

Firstly, you must enter in the general game details. Some of this data is required and some is optional – once again the bold type illustrates required data.



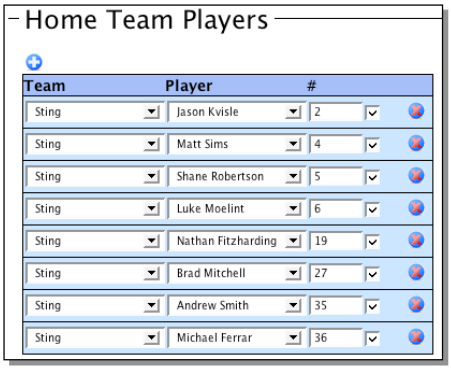
Enter in the actual start and end times of the game as a time in the format: HH:MM. The period durations should be entered as duration in minutes.

Next you need to select which referees, linesmen, and other officials that were involved with the game. If any of the officials are not listed here, you should return to the Officials page and enter in the missing officials. Each of these official lists will be filtered by the officials' functions. An official tagged with the ability to Referee but not Timekeep will consequently appear in the Referee list but not the Linesman list.

Team Results

Next, for each team, you must enter in the details of which players played for each team, the Shots on Goal results, Penalties and Goals. Filling out this data in order will facilitate your workflow and the page is laid out as such. For example, entering in player information will dictate what players are available for other sections such as penalties.

Initially, the players for each team should be filled out for you. These players are a complete list of all the players associated with the given team. For example, if the team Sting were playing as the home team, any player who was registered by default to be playing for Sting would appear in the list. Their default jersey number would also be filled in for you.



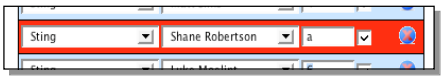
From here, you can edit the players who are in the list. Often, not every player registered with a team will play in the one game. You should then delete those players from the list. You can delete a player by clicking on the delete icon for the corresponding player.

You may also wish to add a player that is not in the list. Occasionally there may be a player from another team, or a player who is not registered with any team, playing for the team. For example, if the Home team was Sting, they may have a player from another team such as Heat playing with them. To add this player, simply click the add icon and select the appropriate team from the team select list. Doing this will change the corresponding list of players from that team. You should now be able to select the player from that team.

For any player, you can also set their jersey number. This may be different from their default jersey number and can be changed at any time.

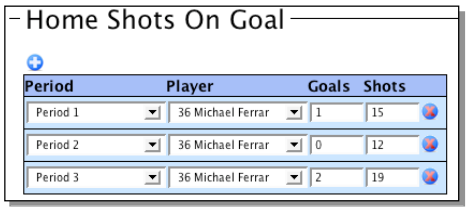
There is also a checkbox to indicate if the player actually played in the game. That is, a player may have been registered to play a game, but they may have sat on the bench the whole game. In this case, you must still register them as part of the game but un-ticking the checkbox will indicate that they did not actually play during the game.

If at any stage during entering a player there is an error with the data, the row in error will appear red. You must fix the row before proceeding.



The next three sections for a team's results are Shots on Goal, Penalties and Goals. Each section is completely dependent on the players' list that has been filled out. For this reason you should fill out the players list first.

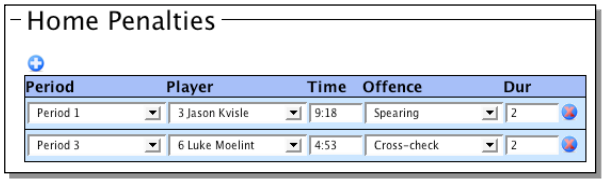
Shots on Goals should be entered as a total of shots on goals and a total of goals for a given goalie in a given period. To add a shots on goal record, simply click on the add icon. Choose the period, and the goalie and enter in the number of goals and the number of shots for that period.



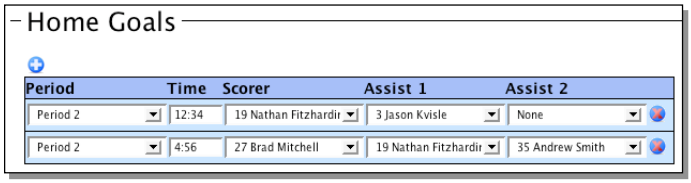
So in the simplest case, you will have 3 entries for a game detailing how many shots on and how many goals were scored for each period against just one goalie.

In a more complex situation where there is multiple goalies playing in the one period, you may wish to record how many shots on and goals were scored against each goalie. In this situation you should have more than one entry for the period.

Next you should enter in any penalties the team has earned. To add a penalty, simply click on the add icon. Choose the period, player and offence. Also enter in the time of the offence in a MM:SS format and the duration of the penalty in minutes.



Next you should enter in any goals the team has scored. To add a goal, click on the add icon. Choose the period, scorer and optionally 1 or 2 assist players and enter in the Time of the goal in MM:SS format.



Lastly, there is room to record any notes about the team or the game.

Once you have completed these results for the home team. Complete the same process for the Away team.

Upon completion of the entire form, click on the Submit Results button. This will firstly run some error checking over the page to verify things like entering in enough players and validating the time formats. Once all the verification is complete, the page will be submitted and the results for the game will be displayed.

Technical Discussion and Implementation Details

When implementing this research project, there were many different technical design decisions that were made. These decisions help illustrate both the micro and macro level of the technologies used in this project.

In general, it was important to develop code that was reusable, functional (or modular), and easy to maintain. Each section of code throughout the site needed to be clearly defined, easy to read, and be included in the right file.

Following is a discussion of some of the more interesting design decisions for each of these technologies.

CSS

When designing the CSS for the project, I decided to split it up into three different files. They were layout.css, form.css and calendar.css.

Layout.css

Firstly, this file was used for the overall layout of the site. It allowed the positioning of the main elements in the site. Those elements were Header, Sidebar, Content and Footer. The layout.css file was also in charge of common formatting across the site. This includes table formatting with alternating row colours, image layout and more. The common formatting was usually quite minimal and did not require its own CSS file.

Form.css

The more complex aspects to the project usually involved XHTML form inputs. These inputs in the past have been traditionally laid out with tables. Nowadays, many people have attempted to layout forms using CSS in an effective and efficient manner. During my research of the various techniques people used, I came across Jeff Howdens site (Howden, 2006). His design was very nice to look at. It was simple and appealing and was implemented without any tables.

My form design was based largely on Howdens work. I adopted his CSS files for use in the Research Project. Unfortunately, his CSS code was very messy. There were elements being controlled by random bits of outdated CSS. I spent a considerable amount of time cleaning up the CSS to be much more simple and specific to my needs. Some of the input forms used in the Research Project, such as the GameDetails page, have many form elements in one row. Howden's code only supported a 2-column approach of label vs control so I added various CSS elements to facilitate these new requirements.

Calendar.css

Lastly, there is the formatting of the calendar used on the Games page. The calendar is based on Googles Calendar. There were various elements of the calendar that were not controlled by CSS. Modifying the CSS and the Calendar JavaScript code, I was able to add various effects like hover formatting and to make the colour scheme of the calendar match the rest of the site.

JavaScript

The Research Project relies heavily on JavaScript for the administration component. In fact, the administration component is not even degradable to run without it. This decision was made very early on. There were far too many features that I wanted to implement to facilitate the administrators workflow. As this component of the site is only going to be used by a few people, placing the restriction that they must use a new browser with JavaScript is a perfectly reasonable constraint.

Due to the heavy use of JavaScript, I found it the following practices helped significantly.

- Firstly, because JavaScript is a prototype language, it is easy to extend existing objects by adding your own functions. In particular, the Array class built into JavaScript 1.5 did not contain various functions that I required. For this reason, I used an array.js to add further functions to the Array class, including the indexOf function that is available in JavaScript 1.6+.

- When dealing with the DOM, it is common to be deleting and creating elements in various different contexts. For this reason, I found it helpful to always pass around an element id rather than passing around a pointer to the element. The reason for this is if during execution of your script, you pass a pointer to an element that gets deleted and another element takes its id, you cannot be guaranteed that you are dealing with the correct element. In fact, you may be dealing with an element that has been deleted and in fact your function is not behaving correctly at all. Working with Ids avoids this problem and also provides more helpful error messages if there is a problem with the script. However, there may be a slight trade off in speed.

- There was a strong focus on reusing scripts and functions. This allows code to be written once and used many times. As the need arises to make the code cross-platform, or when debugging, there is one simple function to debug. A small caveat to reuse in JavaScript is that a JavaScript cannot include or link another JavaScript in directly. It relies on the XHTML page referencing all the dependencies.

- Unfortunately, one of my biggest frustrations with JavaScript is that it is not a type-strict language. While this is really useful at times, I have found it on the whole to be terribly difficult to debug. For example, the interpreter does not even complain if I call a function with a different number of arguments to the signature. Without getting into a discussion comparing type-strict and non strict languages, in practice I found some useful things to help ensure that I was using my functions correctly was to “assert” at the beginning of my functions that all my arguments were valid and also to use strong coding style – such as a_Variable name to indicate arguments, early outs and other common styles.

- In general it is a good idea to write degradable JavaScript, as not all users will have it enabled. The JavaScript in this project however is not completely degradable. This is because the decision to require JavaScript and the latest version of Firefox was made during the design phase.

Apart from the overall approach to JavaScript, there were a couple of specific areas in the project that I used JavaScript heavily. Particularly for form validation and the Game Details page.

Client Side Form validation

When designing the form validation for the project, it became blatantly apparent that the onchange validation and the submission validation were the same. The only difference being that the onchange validation was used for the validation of only one form element where as the submission validation was to validate all the form elements.

For this reason, I needed to write some reusable functions to validate different types of data. With JavaScripts built-in support for regular expressions this was a breeze. I have written functions to validate whether the input is present and valid for types such as numbers, dates and times. As I was researching, I found Stephen Poley’s advice regarding form validation very useful but not entirely applicable (Poley, 2006). While Poley has some great advice in general about JavaScript use in a context of users without constraints on their operating environment, I was able to put constraints on my users within the administration component. However, Poley does provide some other useful tips that I adopted. Examples are using the

onchange method rather than the onblur method for validation as it can cause premature validation, using visual feedback with CSS and a minimal use of alert boxes.

Once the building blocks of validation were in, I needed to hook up the forms with the validation functions. Bearing in mind my desire to write modular reusable code, I designed the system such that the form validation worked by testing the return values of the onchange events for all the form elements in a form. So, when the user clicks the submit button, a script will iterate the form looking for input and select elements that have an onchange function. The results of all of the events are tested and the form is only submitted if all of the tests passed.

Not only does the onchange and submission have validation in common, but also the visual feedback for the two should be the same. The onchange validation provides the visual feedback to the user when their input is in error. Consequentially, when the submission validation calls the onchange events, there will also be visual feedback on any errors at submission time.

The manners in which the errors are displayed are also worth discussing. Again, as reuse was a high priority, I needed to design the system of displaying errors in such a way that both the displayError function would work on different types of elements. I had two contexts that I needed to display visual feedbacks for input errors. The first is the table-less form elements displayed on almost all of the input forms used in the project. The second is to be able to display an error for a particular row in the Game Details input form.

I decided to write the displayError function such that it would take the id of the element that was in error. From there, it would recursively search its parents until it found a parent element that was either a DIV element or a TR element. Once it found one of these elements, it would display the visual feedback accordingly using the same CSS class. Further, by passing only the id of the element that was in error, it allowed the calling code to stay the same and the error function to decide which element to display the error on. The function also takes an actual error message that is displayed as a tool-tip. If in the future, it is decided that a tool-tip is not obvious enough to the user, then the calling code will remain unchanged but how the message is displayed can change.

While the majority of validation is common code, some validation is required which is very specific to certain forms. Specifically there is the validation of a Game Number, on the Games page, which uses AJAX to test if a game number already exists. Another example is the extensive error checking in the Game Details page. Where error checking was specialised it was put in its own corresponding JavaScript file for that page.

Game Details Page

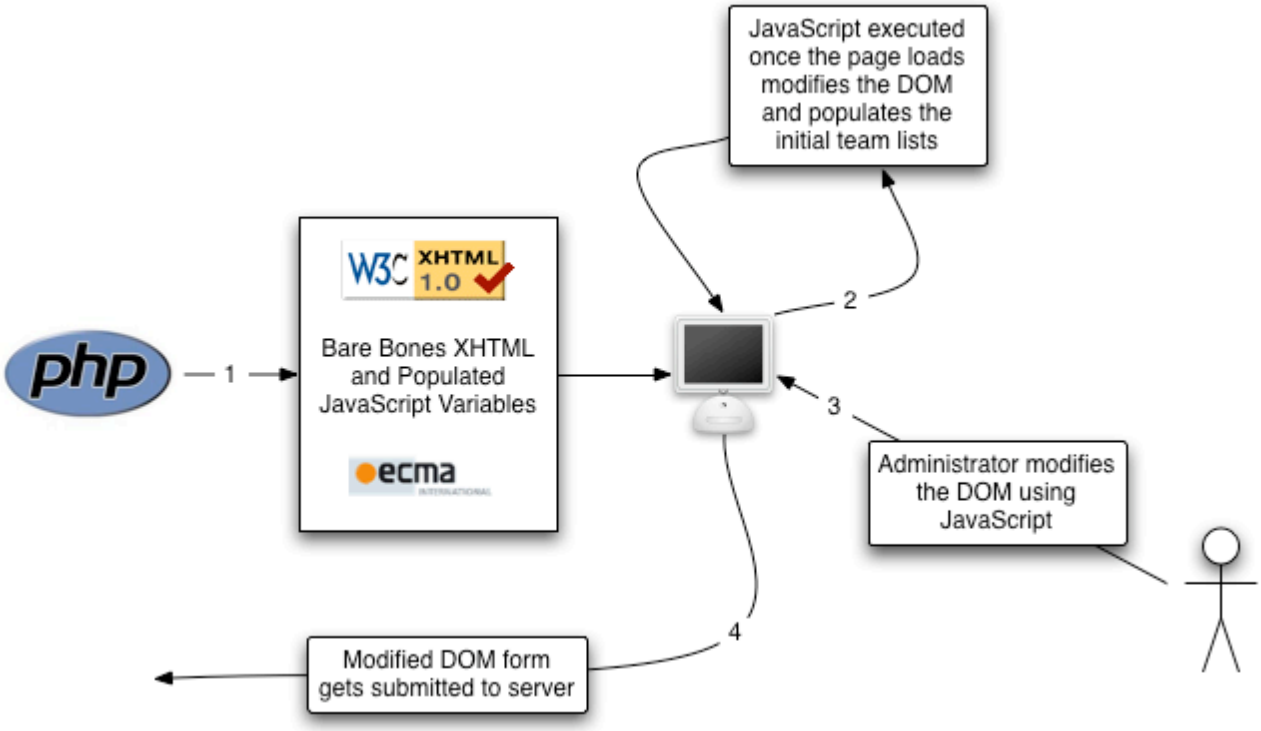
The page that relies upon JavaScript the most is the Game Details page. Not only does it use JavaScript for form validation and CSS modification, it also manipulates the DOM extensively, has cascading on change events and also relies on AJAX. This page represents the culmination of my understanding of client side web technologies. Following is a discussion on some of the interesting features of this page.

DOM Scripting

When entering Game Details, the administrator needs the ability to add and delete players, modify jersey numbers, include players from other teams and more. Various dependencies also need to be tracked and modified. The most common of these is when the Players list is modified; those changes need to be reflected in the player lists in the Shots on Goals, Penalties and Goals sections.

In the past, operations that modify the DOM in such a way would require a round trip to the server in order to build a new web page and return it to the client. Not only that but the state of the users ‘transaction’ would need to be stored and tracked as they made their way through the system. With DOM scripting, the entire operation can be performed at the client end, reducing the communication with the server, refreshing of web pages and complex state tracking.

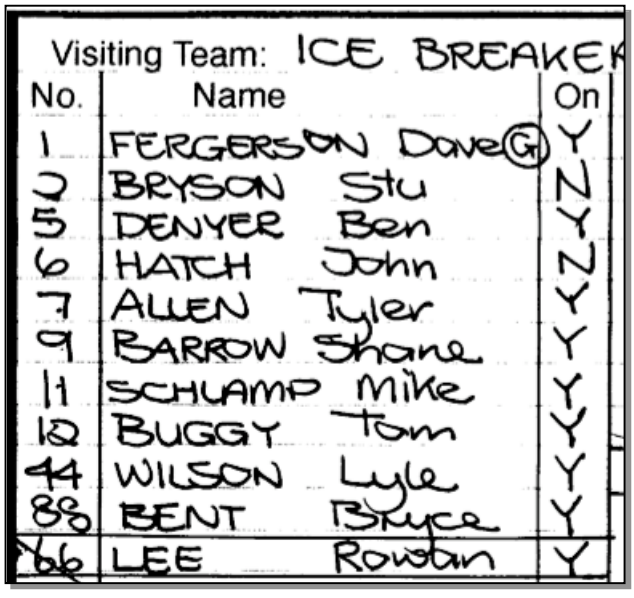
The page is initially constructed in the PHP code as a fairly bare bones page. The top section “General Details” is generated but for the following Home and Away sections, only a placeholder table element is generated for each section – this includes the Players list. Along with the bare bones page, a few global JavaScript variables are populated such that JavaScript can modify the DOM to finish constructing the page. This workflow is illustrated in the following diagram.



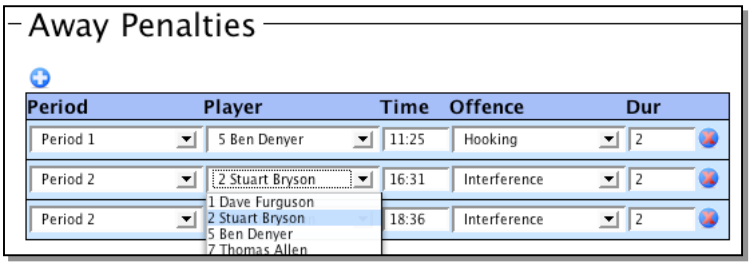
The JavaScript variables that are populated by the PHP code contain a list of teams, offences and period types. These variables are used to populate certain elements in the form. Another global variable that is populated by JavaScript is the player names for both the home and away teams.

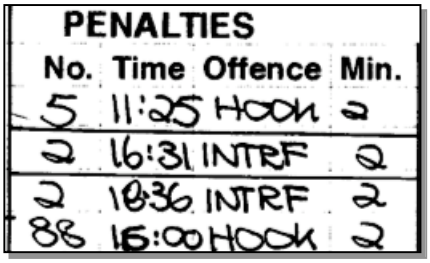
Once the client loads the page, the JavaScript function setupinitialTeamRows is called. This function takes the populated JavaScript player variables and constructs the home and away player lists by adding a row in the table for each player. The player tables were populated using JavaScript instead of PHP so that the code path for both the server generation and the administrator modification of player lists was the same. This reduces code complexity and the chances of bugs with two different code paths.

After the initial team rows have been constructed, the page is presented to the administrator for modification. The first section the administrator must modify, other than the general details section, is the player list for each team. It is important to enter in the team players first as these lists dynamically update the following 3 sections; Shots on Goals, Penalties and Goals. As the player list is modified, the function cacheActivePlayerRows is called. This function iterates the given table and caches a list of players defined by the administrator. This cache is then propagated to the players' lists in the following sections. The following diagrams illustrate entering the players from a game sheet into the game details page.



The player lists found in the sections following the team lists also include the Jersey Number. This is vital as the NSW game sheet only records jersey numbers for Shots on Goals, Penalties and Goals. By providing the jersey numbers at the start of the list, the administrator can use the keyboard to quickly type in the jersey number and have it select the correct player from the list.





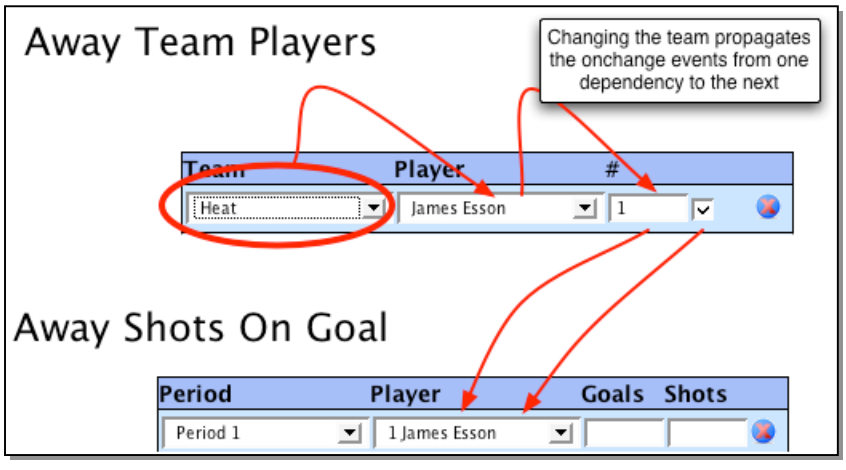
As the administrator adds and removes a row, be it player, penalty or any other type of row, JavaScript is modifying the DOM. For example, to add a row it will query the table to find the next id and append a row to the table. It will then add cells to the row and form elements to each cell as the row type dictates. Again, the code is written modularly so that common functionality is reused between different row types.

Cascading onchange Events

It is not uncommon when designing XHTML forms to find that you need to make selection boxes dependent on each other. That is, if I have two select lists and the values of the second select list vary based on the value of the first select list then the second list is considered dependent on the first. Indeed, this idea of dependent lists was very common when building the game details page. Matt Kruse developed a library to deal with this exact problem (Kruse, 2006). The library called 'Dependent Selects' almost suited my needs. Unfortunately I need something more flexible that allowed me to propagate dependencies to more than one element and also to allow lists to be generated using AJAX and other complex logic.

One example of these dependencies is in the players' section. The administrator must choose the team the player belongs to, then choose the player. The player list only shows the players that are associated with the selected team. As the team changes, the players list changes. The diagram shows the list of players in the Ice Breakers team, and then the list of players available in the Heat team.

Not only does the players' list depend on the team selection, but the jersey number depends on the selected player. Further still the Penalties player list depends on the jersey number. Without a well-designed system, these dependencies could become unmanageable. The system I designed enabled the dependencies to be cascaded such that if at the very top level I changed the Team, all of the dependencies onchange events would be called in dependency order. See the following diagram for an example.



JavaScript Global Variables vs AJAX Data Retrieval

Another interesting feature of the game details page is the use of AJAX to retrieve team player lists. AJAX, however, is not used to retrieve all the team lists, only some. The reason for using AJAX at all is a trade off between what is believed to be absolutely required data vs. potentially required data.

In most cases, players will only play for their default team. Occasionally however, a player may fill in or play the odd game for a different team. So, when the administrator enters in the details for a game, we can safely assume that the required data for the page is going to include a list of players for both the home and away teams. Populating this list of players with PHP into a JavaScript variable is perfectly reasonable.

However, we do not wish to populate this variable with every single player from every single team in the league – imagine if there were 20+ teams. It is highly unlikely the data will ever be used and it will slow the page generation, loading and DOM scripting down. Therefore, instead of populating the JavaScript variable with all of the team lists, we use AJAX to retrieve the team players for a team only when needed. When the administrator selects a team that is not the home or away team, an AJAX request will be sent to retrieve the players for that team. The results of the AJAX request will be stored in the same JavaScript variable that was populated with the home and away team players by PHP. This enables a simple transparent API to the game details page. If the players list for a team_id is in the variable, use that, otherwise send an AJAX request.

CSS is also used to give feedback on the AJAX progress.

PHP

More briefly, the PHP code was also developed in a reusable, modular fashion. Common functionality such as connecting to the database, login verification using MD5 checksums and cookies, creating options lists and sortable headers and verifying the success of MySQL queries were all encapsulated into a common php file.

As mentioned in the System Plan (Bryson, 2006b, p.2), the interaction pattern used for almost all the PHP pages is one of insert, update and delete records. This pattern allowed logical grouping of functionality based around a particular entity.

It is worth mentioning that when designing the PHP code, I considered using PHP templates. They seem to be quite popular in the community (Lozier 2003, Maia 2002). Various different templates include SimpleT, Smarty, and PHPTAL. Some of these templates are fairly basic and others are extremely feature rich. The aim of templates is to abstract the page logic from the presentation. Unfortunately, I was not convinced that PHP templates are any good at actually doing this. It seems the code and presentation, even with the use of templates, do not provide a clean abstraction such as the Model, View, Controller paradigm. I fear that PHP is inherently cursed with never being able to achieve this well. Perhaps another language such as Ruby on Rails or WebObjects is better equipped for this.

Another decision that I made regarding PHP was the use of hidden form elements and using these to set action variables. An action variable is one that tells the PHP code what action you are trying to perform, such as insert or delete a record. Often websites will send the data in post but to a temporary URL and then redirect back to the original page. I did not want this behaviour and instead implemented my actions using hidden form elements. Using the $_REQUEST object, I would check to see if the action post variable was set and act appropriately if it was.

MySQL

MySQL was used for the backend database. Almost all the data was normalised and linked using foreign keys. With the advent of reference constraints in MySQL 5, referential integrity can be maintained at the database level rather than relying on the application logic. Of course, the application logic should still be testing for references. However, there is no longer the concern that buggy code will destroy the integrity of the database. For example, if a player has been entered into a games' details, that player cannot be deleted from the system.

Ajax

As already mentioned, AJAX is used to verify that game numbers are unique and also to retrieve a teams player list. A useful JavaScript AJAX request object written by Matt Kruse (Kruse, 2006) was used to facilitate the use of AJAX.

Future Development

There are many areas for future development, including further client-side error checking, extensive server side error checking, sort direction arrows, running totals on the game details page, sorting lists on the game details page, keyboard shortcuts and better security. This, along with all the extra features detailed in the System Plan.

Conclusion

This Research Project represents a realisation of the all the Learning Objects and the majority of the System Plan. The deviation from the System Plan has been justified. It provides extensive documentation of both the general use and administration components. It provides a detailed specification and technical discussion demonstrating my understanding of 5 core web technologies and their relevance at both the macro and micro level. It is supported by various research and other developers' findings. And lastly, the entire site uses valid XHTML strict and CSS code.

References

SimpleT 2003, <http://simplet.sourceforge.net/>.

PHPTAL 2005, <http://phptal.motion-twin.com/>.

Smarty 2006, <http://smarty.php.net/>.

Bryson, S. 2006, *Learning Contract*, Research Contract, University Technology, Sydney, pp. 1 - 10.

Bryson, S. 2006, *System Plan*, Research Plan, University Technology, Sydney, pp. 1 - 4.

Howden, J. 2006, *CSS Only, Table-less Forms*, Electronic Article, p. 1<http://jeffhowden.com/code/css/forms/>.

Kruse, M. 2006, *Dynamic Option Lists / Dependent Selects*, p. 1<http://www.javascripttoolbox.com/lib/dynamicoptionlist/>.

Lozier, B. 2003, *Template Engines*, p. 1<http://www.massassi.com/php/articles/template_engines/>.

Maia, J.P. 2002, *Introducing Smarty: A PHP Template Engine*, pp. 1 - 3<http://www.onlamp.com/pub/a/php/2002/09/05/smarty.html>.

Poley, S. 2006, *Javascript form validation – doing it right*, p. 1<http://www.xs4all.nl/~sbpoley/webmatters/formval.html>.