****

|  |
| --- |
| National College of Ireland |
| Advanced Rich Internet Applications |

Bodyline Fitness

A Gymnasium Application

Created as a Single Page Application in AngularJS

Lecturer: Adrian Skehilll

John Fitzpatrick (x00113647)

Kean Wei Ho (x12104574)

Ewa Marciniak (x11114533)

Thomas Dowling (x12113441)

May 2nd, 2014

|  |  |
| --- | --- |
| **Github** | [github.com/ewamarciniak/advanced\_rich\_internet\_applications](file:///C:\Users\md\AppData\Roaming\Microsoft\Word\github.com\ewamarciniak\advanced_rich_internet_applications) |
| **Deployment** | http://ewamarciniak.github.io/advanced\_rich\_internet\_applications/app/ |

Table of Contents

[2 Introduction 3](#_Toc386984757)

[3 Motivation 3](#_Toc386984758)

[3.1 Project scope 3](#_Toc386984759)

[3.2 Area of Contribution 3](#_Toc386984760)

[4 State of the Art Review 4](#_Toc386984761)

[4.1 Technologies Used 4](#_Toc386984762)

[4.2 AngularJS 4](#_Toc386984763)

[4.3 Protractor 5](#_Toc386984764)

[4.4 SkelJS 5](#_Toc386984765)

[4.5 Canvas.JS 6](#_Toc386984766)

[4.6 Google Calendar API 6](#_Toc386984767)

[4.7 Google Map API 7](#_Toc386984768)

[4.8 Critical Review of Strategy Adopted 8](#_Toc386984769)

[4.8.1 Possibilities Considered 8](#_Toc386984770)

[4.8.2 Approach Adopted 8](#_Toc386984771)

[4.8.3 Methods and Strategy 8](#_Toc386984772)

[4.8.4 Critical Analysis 9](#_Toc386984773)

[5 User Interface Design 9](#_Toc386984774)

[6 Architecture 11](#_Toc386984775)

[6.1 Application Architecture 11](#_Toc386984776)

[6.2 Security 13](#_Toc386984777)

[6.2.1 Skel and Angular as a security risk 13](#_Toc386984778)

[6.2.2 Angular defence against security risks 14](#_Toc386984779)

[6.3 Data Transfer Strategies 14](#_Toc386984780)

[6.4 Evaluation and Testing 15](#_Toc386984781)

[7 Summary 16](#_Toc386984782)

[8 Bibliography 17](#_Toc386984783)

# Introduction

The aim of this project was to create a Rich Internet Application for a gymnasium enterprise that would be available on-line, viewable on any device connected to the Internet (mobile phone, PC or tablet), and platform-independent. An additional goal was to create an application which had ‘the look and feel’ of a desktop application using only modern open-source technologies. A further aim was to ensure that the application adhered to the principles of responsive design and test-driven development.

# Motivation

## Project scope

The Application was created as a single page application with the AngularJS framework, together with HTML-5, CSS3, jQuery and JavaScript. Responsive design was achieved using the SkelJS framework. Testing was done with Protractor, an-end-to-end test framework for [AngularJS](http://angularjs.org/) (<https://github.com/angular/protractor>), where the testing framework used by Protractor was Jasmin (http://jasmine.github.io/). Version control was with Github, and the application was deployed using Github Pages.

This combination of technologies allowed the creation of an application using a Model-View-Controller architecture where the presentation layer and all of domain logic lies on the client,

## Area of Contribution

It was decided to create a Rich Internet Application for a gymnasium enterprise where the following functionality would be incorporated (see [Fig. 1](#figOneBOdyline) for screenshot of the home page).

* An attractive home page
* Dynamic search functionality
* Find a gym location using Google Maps
* View Workout statistics
* Calendar

A desirable feature of such an application is that page load times are minimized, as a gymnasium user would be expected to want to browse quickly to the required information and functionality. This is a further advantage of using the Angular JS framework: all code is incorporated into a single page where partial HTML files are loaded asynchronously and ‘injected’ into the single HTML page (index.html) without the need for a new HTTP request.

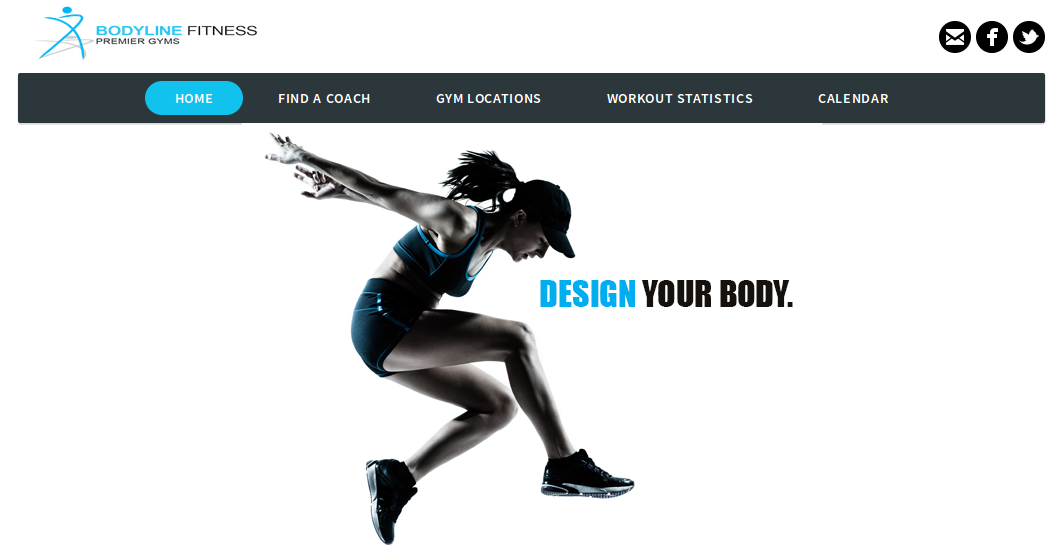


Fig. 1. A Screenshot of the Homepage of Bodyline Fitness.

# State of the Art Review

## Technologies Used

The principal technologies used were the following:

1. AngularJS <https://angularjs.org/>
2. Protractor <https://github.com/angular/protractor>
3. SkelJS http://skeljs.org/
4. CanvasJS http://canvasjs.com/
5. Google Calendar API <https://developers.google.com/google-apps/ca>
6. Google Map API https://developers.google.com/maps/

## AngularJS

AngularJS[[1]](#footnote-1) is an open-source JavaScript framework created by Google and first released in May 2011 (Hevery & Green, 2014). It allows dynamic functionality to be added to HTML using a very powerful syntax and Model-View Controller architecture. It features data binding and dependency injection. Data binding may be thought of as method of changing **the view** whenever the **model** changes. **Controllers** are written in JavaScript and control the behaviour of DOM elements. To quote directly from the Angular Website: ‘AngularJS lets you express the behaviour in a clean readable form without the usual boilerplate of updating the DOM, registering callbacks or watching model changes’ (Angular, 2014).

Web apps build in AngularJS are well structured, testable, and maintainable front-end applications. In layman terms, AngularJS is a Framework and HTML compiler. With AngularJS, you tends to “think –architecture”, and single-page applications (SPAs) that are light-weight, testable and easily deployed may be quickly and efficiently created.

Advantages

1. A lightweight, intuitive open source framework that allows dynamic functionality to be incorporated into HTML documents with an intuitive syntax.
2. Uses a MVC architecture and separation of concerns is easy to implement.
3. Easy to test functionality
4. Widely uses and very well documented.
5. Easy deployment.

Disadvantages

1. Angular Controllers are not compatible with jQuery.
2. Angular controls are incorporated into the HTML mark-up, and violates the principle of unobtrusive JavaScript (Ullman)

## Protractor

Protractor is a powerful testing framework for AngularJS which allows end-to-end testing. It is built on top [WebDriverJS](https://code.google.com/p/selenium/wiki/WebDriverJs), and tests an App by running it in a Browser. The Jasmin framework ([jasmine.github.io](http://jasmine.github.io)) and syntax may be used to write the tests.

Advantages

1. Allows the actual functional website to be tested.
2. Any piece of functionality is potentially testable
3. Tests may be written in Jasmin syntax.

Disadvantage

1. Can be slow. It may take several minutes for an App to be tested.

## SkelJS

To quote directly from the SkelJS website, “SkelJS is a lightweight frontend framework for building responsive sites and apps“(Skel.org, 2014). All of the code is contained in a single JavaScript file, which for the present release is 20 Kb.

There are four powerful components (Skel.org, 2014).

1. CSS 12 column grid system

A 12 column CSS grid system offers unlimited nesting and adjustable gutters, among many other features.

1. Responsive Handler

CSS media queries may be used to assign a different stylesheet depending on the browser window style (see, for example, <http://css-tricks.com/css-media-queries/>).

The Responsive Handler is a simple, easy-to-configure replacement for CSS media queries which can handle any number of breakpoints and, as the SkelJS website has put it, “ [is] capable of doing much more than simply shuffle style sheets” (Slel.org, 2014).

1. CSS Shortcuts

Useful shortcuts for common CSS tasks

1. Plugin System

SkelJS provides a platform for installing plugins without compromising its lightweight status.

Advantages

1. Allows responsive design functionality to be easily incorporated into an App.
2. Lightweight
3. Open source

## Canvas.JS

CanvasJS is a HTML/JS charting library built on top of the HTML-5 Canvas element that allows attractive, dynamic charts and graphs to be constructed. Its many features include preloaded themes, cross-platform compatibility and high performance. CanvasJS can render 105 data points in less than 100 ms. It is free to non-profit organizations, but use in commercial applications must be paid for (Canvasjs.com, 2014).

## Google Calendar API

The Google Calendar API allows a calendar to be incorporated into a web application where a user may create new events and edit and delete existing events.

Distinctive Features

1. Calendar events can have information like title of event, start and end times, and attendees.
2. A lists of calendars in the Calendar UI
3. Display options from user preferences in the Calendar UI, such as user’s time zone.
4. A list of colour to choose for the events and the calendars.
5. A feature that show that you are Free/Busy in the set of time or set of calendars.

Advantages

1. Save a lot of time and money. APIs are free make it easy to use huge infrastructures instantly.
2. Potential for innovative ideas. For example can integrate Google Calendar with minimum costs.

Disadvantages

1. Dependence on external services. If the service or the company shuts down so that the API is no longer maintained or is removed completely, this will adversely affect the dependent website.
2. Too many APIs on a website can be confusing as because API providers do not adhered to a common standard.

The above information was taken from the following sources: Google Developers, 2014a,b

## Google Map API

Google Maps offer a wide range of APIs that allows Google Maps to be embedded into an application. This includes the Google Map [JavaScript API](https://developers.google.com/maps/documentation/javascript/tutorial) which allows an interactive Google Map to be incorporated.

Distinctive Features:

1. Free API key from Google.
2. Creating a basic Google Map is an easy task.
3. Map overlays has options where you can add a marker in the Google Map.
4. Add Maps Events so that users can click the marker to zoom and attached event handlers to Google Map.
5. Google Maps Controls that with the default set like Zoom, Pan, map type, and street view.
6. There are 4 types of google maps available: Roadmap, Satellite, Hybrid, and Terrain.

Advantages:

1. Most APIs are free or very cheap.
2. Leveraging the resources of the web could save a lot of time in terms of development Time.

Disadvantages:

1. The reliability of the system is depends on the provider, since it is free, the provider do not have the responsibility to keep your app operational.

Source: Google developers (2014b).

This section should outline a literature review of the methodologies.

Describe investigation into current practice, case studies of other rich Internet applications, and academic/industrial research that is reported in conferences and journal articles. This should include a critical analysis of your selected approach. You must describe how you made an informed decision about both the innovation undertaken and the technologies selected.

## Critical Review of Strategy Adopted

### Possibilities Considered

At an initial meeting of the team, various possibilities were considered.

1. Build a ‘back-end’ with Ruby-on-Rails, and develop a Rich Internet Application ‘front-end’ on top.
2. Use Microsoft Visual Studio 2013 for the ‘back-end’ with (as in 1) a Rich Internet Application ‘front end’.
3. Use PHP for the ‘back-end’ and develop the client-side with HTML5, jQuery and JavaScript.
4. Build a Single Page Application with AngularJS where data is read from JSON files.

### Approach Adopted

The main criticism of Ruby-on-Rails was that it is too difficult to deploy. Visual Studio was rejected in favour of open-source alternatives. An initial attempt to use PHP within an AngularJS App lead to difficulties and this approach was abandoned.

It was decided to ‘try something new’ and to attempt to build a single page application using AngularJS. This is an open source framework that uses and extends a programming language (JavaScript) that is fast becoming the language of the web. The group members had no previous experience of using MVC exclusively within the presentation layer (client-side), and it was felt that it would be a good challenge. (All members of the group had experience of MVC with Ruby-on-Rails). AngularJS is also extremely popular, well documented, easy to test, easy to deploy, and lightweight.

### Methods and Strategy

The following is a summary of the development process

1. A github account was established for version control.

https://github.com/ewamarciniak/advanced\_rich\_internet\_applications

1. An skeleton Single Page Application was developed and deployed on Github pages
2. SkelJS functionality was added
3. Search, CanvasJS, Google Maps API Google Calendar API was developed and incorporated into the App
4. Functionality to allow data to read from JSON files was incorporated.
5. Tests were written to allow end-to-end testing with Protractor
6. An attractive UI was developed
7. The App was deployed on Github pages (Test functionality was not deployed).

<http://ewamarciniak.github.io/advanced_rich_internet_applications/app/#/view1>

### Critical Analysis

Once an initial learning curve had been overcome, App development with AngularJS was very successful, allowing expression in code of the ideas and aspirations of the group. All of the technologies used integrated very well: there were almost no conflicts. An exception is jQuery which is not very compatible with AngularJS, and it was generally felt that this was a disadvantage.

The AngularJS implementation of the MVC architecture allowed for easy code development and integration. Each partial had its own separate controller and each member of the team could concentrate on a particular area governed by a single partial and controller. It was then very easy to integrate everything into the fully functional Application.

Perhaps the most impressive aspect of AngularJS is that a Single Page Application was built without the developer having to implement any Ajax functionality.

Deployment to Github pages was very easy and required no special modification to the code base. All that is required is that a separate Github branch is created (git checkout –origin gh-pages), add the code, and then ‘push’ (git push origin gh-pages). This is another major advantage of the AngularJS approach. The only major disadvantage is that the powerful ‘back-end’ functionality provided by (say) a Ruby-on-Rails application is absent.

# User Interface Design

The main goal when deigning the User Interface (UI) for the project was to make the user interaction simple, intuitive and efficient. The following section describes the process employed by the team to arrive at the design of the UI.

The formulation of the UI design was influenced by a number of factors.

At the start of the project the group researched and discussed the requirements of the potential users. Due to the tight time constraints and limited budget, the project requirements were finalized based on the research, rather than a full scale functionality requirement gathering techniques such as interview or questionnaires. It was agreed that this application must be simple and functional and must deliver specific gym information to its members outside of the gym. This list included a list of gym locations nearby, class schedules, workout statistics, etc. Users tend to search for the nearby gym location when traveling, check the class schedule on the way back home, and check their own workout statistics when on the bus. This sort of content would normally be accessed by the users “on the go” and at various locations outside the gym, not only on PCs but also tablets and mobiles.

In the era of growing popularity of smart mobile devices and easy access to the Internet, the importance of a mobile-friendly design cannot be understated. Creating a number of separate applications specific to each type of device was never an option as it would not be sustainable in the long term to maintain such a large number of applications. Also due to the tight time constraints this was not possible. The chosen solution was to implement a responsive design approach that would provide a unified user experience throughout the site across multiple devices, and for a number of different screen sizes.

A responsive design can be implemented quite efficiently using one of the many JavaScript frameworks such as SkelJS or Foundation. Due to the level of skill and experience recently acquired by the members during this course module, we decided to use SkelJS.

“*SkelJS is a lightweight frontend framework (…) consisting of only a single JS file - weighing in at just 20kb as of this version*.” (Skeljs.org, 2014).

It is characterized by a sophisticated 12-column CSS [grid](http://www.jqueryscript.net/tags.php?/grid/) system which allows proportional scaling of UI elements and includes a breakpoint handler which intelligently replaces the CSS queries depending on the type of media the app is displayed on. SkelJS was very easy to implement into the app.

Another factor that influenced the UI design was the growing user-experience expectations.“*People are beginning to demand a higher standard of user experience (UX) quality from the software systems they use in their business and personal lives.”* (Anderson, McRee and Wilson, 2010). Single page applications (SPA) with desktop like behavior implemented into web based applications are rapidly growing in popularity and slowly becoming the new standard. Users no longer are required to wait for the page to reload and instead are presented with the data almost instantaneously.

In order to oblige with the current expectations the project was designed and implemented as a SPA configuration using AngularJS. AngularJS is a powerful JavaScript framework that applies the MVC structure to the project’s front-end business logic and provides a number of comprehensive features such as dependency injection, custom HTML templates, and two way binding. The implementation of this framework did not shape as much the form of the interface, as it improved the user experience. When selecting menu tabs, users are seamlessly taken from one page to another without the page load waiting time.

Based on the established requirements the team concentrated on forming the information architecture. Due to the relatively simple nature of the app, it has been decided that the navigational structure will be flat and will consist of a 5 tabs, each corresponding to a separate section. Due to the personal and confidential aspect of the workout statistics displayed on one of the pages the team decided to implement a simple authorization system. Unfortunately due to the problematic nature of back end implementation and limited time constraints this was later abandoned. The login Controls initially added to the interface were consequently removed.

One of the goals of the user interface design was to implement both usability and accessibility principles into the application. “*Usability is about designing products to be effective, efficient, and satisfying”* and its key aspect is creating positive user experience. “*The goal of web accessibility is to make the Web work well for people, specifically people with disabilities”* (W3.org, 2014). Combining both usability and accessibility meant creating both satisfying and disabled-people-friendly applications. SPA implementation was a definite step towards the improved user experience and increased usability. Additionally the graphic design was supposed to serve the purpose of enhancing both usability and accessibility.

The simplicity of the structure and design was meant to enhance both the accessibility and usability of the application. Additionally the design incorporates the following features:

* Provision of text alternatives for any non-text content
* Minimal amount of color to prevent difficulties for people with color blindness
* Use of relatively large font and use of highly contrasting colors (black and white) on the text and background to help people with poor eyesight
* Images contain the alt values (Heng, 2014)
* Consistency in the design

The team aspired to design the User interface to the best principle and standards, as well as to provide a high aesthetic appeal of the design at the same time balancing the visual and functional aspects of the system. Although the design could be further improved we feel that we have achieved some success in fulfilling these aspirations.

# Architecture

## Application Architecture

The architecture of the Advanced Internet Project is a set of layered technologies wrapped inside a top layer to produce a stable responsive user interface.

The layers are designed to give as much code control to the programmer in order to allow for efficiency and structure to the overall flow of code.

The layers are specified as follows:

1. SkelJS
2. AngularJS
3. AngularJS Partials
4. Angular Controllers
5. CanvasJS
6. HTML and CSS Integration
7. JavaScript Control within Angular

The layers are linked or wrapped in depending on which technology depends on another. A diagram of these structures will help to explain this idea ([Fig. 2](#Fig2JohnMain)).

SKEL

Angular

Partial1

Partial2

Partial3

Partial4

Partial5

Partial6

Partial....n

Ctrl1

Ctrl2

Ctrl3

Ctrl4

Ctrl5

Ctrl6

Ctrl....n

JavaScript Functions

CSS

Html Page

*return*

**Fig. 2.**

Each of the layers above has a precedence of action which are defined by the arrows.

The architecture operation and behaviours.

Skel-JS is a framework that reacts to the size and with of the container that it is defined in; in this case the Browser is the container; therefore, the design of the Web Application is based on a responsive site that is compatible with mobile or Tablet devices as a mobile app

In the ARIA project the SkelJS framework acts as a wrapper around the Angular toolset so that any development of an Angular framework will behave responsively because of the SkelJS container.

The Angular toolset is programmed to act as a single page document that can have layers of partial Html pages placed within the main Angular page.

Each Partial Html page is called from the main Angular Menu system which is then displayed as one Web page of Information.

The Partial Pages can have a dependency on Angular Controllers which are a composite of JavaScript, JQuery and other Script.

These Scripts use the controllers for sending or calling functions and accepting a returning value or array values.

The Angular system process Web pages and uses a REST type methodology for navigation using a designated naming convention such as /View1.../View2...etc.

A typical Url would have the format of

Protocol/Host/Path/#/View1

The Partial loads as View1 or whichever designated word description assigned to the partial with in the code. See Fig2.

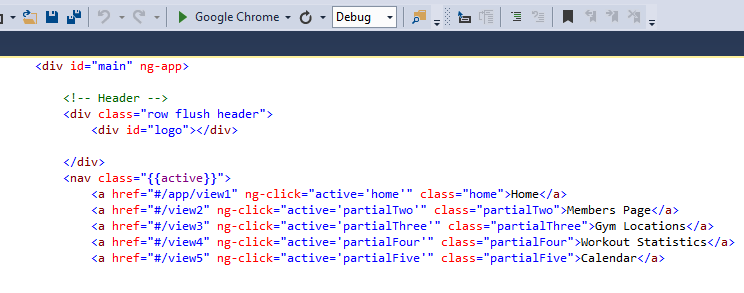


Fig 3.

How these layers are wrapped is shown in Fig. 4.

Skel-JS (responsive)

Angular

Partials (HTML)

Ctrl (JavaScript)

css /jquery/

Fig 4.

Figure 4 shows the skelJS wrapping as the Angular code is placed within the SkelJS container in code. SkelJS is the topmost layer and therefore all Angular code and dependency toolset code, i.e. Partial HTML, Partial JavaScript act responsively with the top SkelJS Layer.

## Security

### Skel and Angular as a security risk

The Internet can be a wide open network and security is a major issue particularly with client-side scripting.

The Angular Framework and JavaScript is always vulnerable because of the nature of client-Side Scripting and because of executables that can be embedded into otherwise static web content.

As a major defence against security, JavaScript does not support the

read/write to local files nor can it list files on a Client PC. So as Angular and SkelJS have a control dependency on JavaScript, they do not pose a Security risk. JavaScript cannot use a Clients computer as a platform to attack or attempt a password crack on other networks.

### Angular defence against security risks

Even though a basic level of security issues are adopted by JavaScript, there are more issues that can still exist.

The main concern is to stop JavaScript programs from exporting private information about or on the user.

Angular is implemented on a level in which AJAX based client framework. Because of this, a user (hacker) can potentially bypass every security measure that is implement on the client side. Therefore all security is implemented on the SERVER-SIDE. So, on the server-side authentication is necessary, so that access to any resource is authorised.

So as a rule, data rendering is not done on the client side as a security precaution.

Unauthorised AngularJS Ajax requests respond with a 401 that is handled by the client accordingly.

Issues with Angular Security that are addressed.

* Vulnerability of all AngularJS framework apps
* Ajax and JavaScript openness.
* The Object is Global in a Browser.
* Scripts can interact with same objects. Variables can be overwritten and Functions redefined.
* Http request can be initiated (using GET and POST)
* Cookie Information can be obtained.
* XSS Attacks

## Data Transfer Strategies

Data was read directly from JSON files from an Angular Controller (/app/controllers.js).

For example, to make the data in gymCoaches.json available to the application (partial 2), a HTTP request to the file is made from within the relevant controller (myCtrl2) as follows:

.controller('MyCtrl2', function($scope, $http) {

$http.get('./gymCoaches.json').success(function(data){

$scope.coachList = data;

});

}

})

The data are now available and may be incorporated into the HTML as follows:

<li ng-repeat="coach in coachList>

<span> {{coach.name}}</span>

...

</li>

## Evaluation and Testing

Testing was done with Protractor, an-end-to-end test framework for [AngularJS](http://angularjs.org/) (<https://github.com/angular/protractor>). This allows end-to-end testing of an App where the tests are run on the actual website.

The framework requires that node.js (http://nodejs.org/) and the Selenium WebDriver be installed (<https://code.google.com/p/selenium/wiki/WebDriverJs>).

The approach is probably best described by way of an example.

Within *partial3.html* there is a button that calls a function, *jourz().* The relevant HTML is as follows:

<input id="Button1" type="button" value="Find" class = "button" ng-click="jourz()" ng-model="name"/>

When *jourz()* is called (defined within controller 3), a gym location is displayed dynamically depending on the value chosen by the user in a drop-down menu.

It is required to test this function. The approach in pseudo-code is as follows. Tell the webdriver to go partial3.html, select (say) option 2, click on the button, and compare the result obtained with the expected result.

The relevant test code (/test/e2e/scenarios.js) is as follows where ‘*addressinput*’ is the id of the relevant input box:

it('should show correct text for gym location', function () {

var ptor = protractor.getInstance();

//Define a protractor instance

ptor.findElement(protractor.By.css('#addressinput option:nth-child(2)')).click();

//Find the drop-down menu and select option 2

ptor.findElement(protractor.By.css('#Button1')).click();

// Click on the button

expect(ptor.findElement(protractor.By.css('#l1')).getText()).toMatch('Quarry Road');

//Compare the result with the expected result

});

In order to run the test (from an Ubuntu system) only the following commands need be executed.

1. npm start
2. protractor ./test/protractor-conf.js

In the considered opinion of the team, this is an excellent way to implement test-driven development. The major advantage is that the tests are being run on the actual website. One disadvantage is that is proved very difficult to define tests to check if content such as Goole Maps was loaded. However, presumable this is do-able if time permitted.

# Summary

A Single Page Application has been developed and deployed for a gymnasium enterprise (Bodyline). A user may search for a gym coach using dynamic search functionality, may view the location of a gym on Google Maps, may view exercise and workout statistics (CanvasJS), or may record an event using Google calendar.

The App is responsive (SkelJS) and was developed in accordance with the principles of test-driven development (Protractor).

The site has been deployed on Github pages

<http://ewamarciniak.github.io/advanced_rich_internet_applications/app/>

# Bibliography

Anderson, J., McRee, J. and Wilson, R. (2010). Effective UI. 1st edn. O'Reilly, Beijing.

AngularJS (2014). What is Angular? [Internet]. Accessed 1st May 2014. Available at: https://docs.angularjs.org/guide/introduction

Hevery, M. & Green, B. (2014). Keynote speech NG-Conference 2014. [Internet]. Accessed May 1st 2014. Available at (video): https://www.youtube.com/watch?v=r1A1VR0ibIQ

Freeman, A. (2012) *Pro ASP.NET MVC 4.*  Apres.

Google Developers (2014a) Google Developers Blog [Internet]. Accessed May 2nd 2014. Available at: http://googledevelopers.blogspot.ie/

Google Developers (2014b) Inspiring developers everywhere. [Internet]. Available at: https://developers.google.com/

Heng, C. (2014). *How to Improve the Accessibility of Your Website (thesitewizard.com)*. [Internet]. Accessed 30 Apr. 2014. Available at: <http://www.thesitewizard.com/webdesign/improve-accessibility.shtml>

Skeljs.org (2014). SkelJS: Documentation. [Internet]. Accessed 28th April, 2014. Available at: <http://skeljs.org/docs>.

Ullman, L. (2012) Modern JavaScript. Develop and Design 2nd edn. PeachPit Press.

W3.org (2014). *Web Accessibility and Usability Working Together*. [Internet] Available at: http://www.w3.org/WAI/intro/usable [Accessed 30 Apr. 2014].

1. Why the name ‘Angular’? To directly quote Miško Hevery, a leading member of the AngularJS development team. ‘Because it [Angular] extends HTML, and HTML uses angular brackets’ (Hevery & Green, 2014). [↑](#footnote-ref-1)