National College of Ireland

**Advanced Rich Internet Applications**

**AriaQuen T-shirt Design Store**

Technical Report

May 6th 2014

Submitted By:

Sandra McCormack x12110639

Miguel Angel Castillo Garcia x13104322

Keith Michael Thomas Gregg x12114090

1. What is a RIA? 2

1.1 Project Scope 3

2. Motivation 4

2.1 Area of contribution 4

2.1.1 Problem Domain 4

3. State of the Art Review 5

3.1 Frameworks, Toolkits & Solution Architecture 5

3.1.1 History of RIA - Flash, Silverlight and Flex 5

4. User Interface Design 22

4.1 Requirements – Functional and Non-Functional 22

4.2 Considerations 23

4.3 Responsiveness 23

5. Application Architecture 25

5.1 Development and Analysis Methods 25

5.2 Software Process 25

5.3 Security 26

5.4 Error Handling 26

5.5 Toolkits and Frameworks 26

5.6 Data Transfer Strategies 28

Advanced Product Search 28

Flickr Gallery 29

5.7 Evaluation and Testing 31

6. Summary 34

References 38

# 

# 1. What is a RIA?

Rich Internet Applications have had an enormous impact on all corners of the world wide web. RIAs offer rich, graphical UIs and a more dynamic user experience than the static pages of Web 1.0 that came before them. Web 2.0 places a greater emphasis on user interactivity, participation and generation of content, with the average person becoming the biggest contributor of material online. Both blogs and social media are key features of Web 2.0, along with tools such as RSS feeds, video casting, social bookmarking, podcasts and picture sharing.

Desktop-like performance is one of the main characteristics of RIA and Google Drive is a good example of this Software as a Service functionality. The user benefits from all the functionality of desktop applications for creating documents, spreadsheets and presentations, etc. yet this is all hosted on the web. The addition of this functionality is possible through the utilization of advanced client side scripting technologies including JavaScript, jQuery, JSON and Ajax.

## 1.1 Project Scope

In brief, this report details the development methodology, technical approach, aims and process used for the AriaQuen e-commerce web application. This document outlines in greater detail the research we undertook, optimization and security considerations and what technologies were utilized in this project. The reasons for their use will be discussed, along with the ways in which we applied these technologies, advantages and disadvantages of their use and further improvements that could be applied to the application in the future.

Our team chose to build an e-commerce web application for the fictional brand AriaQuen. The site provides a new way in which to design and sell clothing. Users can upload their own designs and have them printed onto a garment of their choice. The AriaQuen store stocks a variety of clothing from brands such as American Apparel and Fruit of the Loom, including t-shirts, dresses and sweaters for men, women and kids. The store specializes in fast fashion with a minimalistic, contemporary aesthetic and has a target customer base of design-conscious, trend-savvy men and women.

It is interactive and dynamic, with a large amount of image content as it must appeal to visual people who are interested in fashion and shopping. The performance of the site was key in this development, with the aim of providing an excellent user experience. The primary objective overall was to demonstrate interactions and behaviors which are not possible without the use of rich client side technologies.

# 2. Motivation

## 2.1 Area of contribution

Most online t-shirt design companies (eg. Qwertee, TeeBusters) do not allow you to have rich drag and drop design on the fly, using a much more rigorous download and edit, repost method. By using AriaQuen, we will place an emphasis on design-on-the-fly, using HTML5 Canvas and CSS3 Transformations.

### 2.1.1 Problem Domain

*"The web is the ultimate customer-empowering environment. He or she who clicks the mouse gets to decide everything. It is so easy to go elsewhere; all the competitors in the world are but a mouseclick away."* (Nielsen, 1999)

Web shoppers are becoming increasingly sophisticated as e-commerce and design practices have evolved. Users now expect sites to provide a dynamic, interactive and transparent shopping experience, along with excellent value and strong brand identity. Business owners and web developers can capitalize on providing the best user experience. E-commerce applications are often overloaded with information for customers such as product images, descriptions, prices, customer reviews, branding, promotions and navigation. Users can easily become visually overwhelmed and find sites difficult to use. Basic layout and design principles can be applied for ease of use, without sacrificing content.

*“The experience of effortlessness is a fundamental goal of successful site design and a key pillar of a successful user experience – one that sums up decades of user research into two dimensions”.* (Goddard, 2008)

Client side scripting technologies are designed to manipulate elements on the page and display content that is returned from the server. It allows the site to react to user’s actions by hiding or displaying content, moving elements around or changing styles. Examples of this include expandable navigation items and sliding images. Server-side scripts are not suited to delivering this kind of content as it causes the interface to become sluggish and increase the amount of bandwidth required.

# 3. State of the Art Review

## 3.1 Frameworks, Toolkits & Solution Architecture

For this application our team utilised Sass and CSS3, HTML5, jQuery, jQuery UI, Ajax, JSON and a Flickr API to provide a dynamic, exciting user experience with a wide range of functionality. Ajax requests would not have been possible without the use of a server side framework. Although our team did not branch out further into other areas like AngularJS or EmberJS we would like to explore these technologies further in the future. The main library used to enhance the application was jQuery and we will describe how this was applied in much greater detail.

### 3.1.1 History of RIA - Flash, Silverlight and Flex, dead as a dodo

Before HTML5 and CSS3, Browsers supported different features, and no matter what you did, it was difficult to ensure cross browser compatibility. With Flash, designers and developers were able to create a cross platform cross browser Rich Internet Application, relying on plugins to deliver rich content. The downside to this was that Flash (also Silverlight and Flex) relied on many plugins, file sizes were overly large, and consumed a lot of power and processing capabilities, which were just not suited to mobile and ubiquitous devices. Flash was no longer the one size fits all go to it had been. Online services such as youtube, which delivered content via flash, looked to move to HTML5 video.

#### Apple says No

Apple launched the iPhone in 2008. It did not support flash, which was the industry standard at that time. In doing so, and loading its support behind the lighter, flexible HTML5, it knocked Flash and adobe from the mantle of cross browser RIA king, and opened up the rapid development and acceptance of HTML5 standards.

#### 3.1.2 Future of RIA

##### HTML

HTML5 has many advancements built into its spec. It utilises many APIs such as geolocation to enhance semantic web features. It perfectly marries behaviour, presentation and content, more more so that XHTML did.

##### CSS

CSS3 makes the site’s behaviour lightweight, it allows light and fast functionality when compared to heavier JavaScript. In the project we utilised CSS3 Galleries, Lightbox, and alertify.js CSS pop ups, making code less obtrusive. It also makes response times faster, and server calls lighter, meaning more robust applications (so long as you don’t use internet exploder). We also used Sass CSS, which stands for Syntactically Awesome Style Sheets. It allows for modular, dynamic CSS within the application.

##### JavaScript

Asynchronous Data Transfer was used in the application. This is covered in greater detail in the Data Transfer Strategies section, which can be found below.

#### 3.1.3 Innovation in HTML5

HTML5 extends on HTML4 and XHTML. It has several new features, and deprecates many older features no longer relevant in semantic web developement. It allows for JavaScript API attributes to add additional content, such as Canvas calling functions, HTML5 Video and Audio natively in the browser etc.

#### 3.1.4 Innovation in CSS3

A lot of what was previously only do-able in JavaScript can now be implemented using CSS3 techniques. With the death of [IE6 and IE7 in recent months](http://www.ie6death.com/), newer web standards are now supported in more contemporary, up-to-date browsers. JavaScript can now focus on what is does best, behaviour and data management/binding (JavaScript everywhere with node.js) and CSS can look after presentation and display of UI.

You may ask why would one use CSS3 over JavaScript for content like lightboxes, pop-ups etc.. The main reason would be that CSS3 is *faster* and *lighter*  than its JavaScript version. it also relies on less processing power, making sites quicker and more efficient.

The following features were included in the application:

##### Parallax Scrolling

Description

Parallax scrolling homepage/splash page gives a taster of the application. It incorporates a variety of JavaScript code as well as using CSS3 transitions. The idea behind parallax is that is creates a visual stimulus to engage the user in a dynamic and engaging way. By adding to the basic parallax code, incorporating mini-games at certain points, as well as a mini gallery and hot points, we have attempted to grab the users attention from the get-go.

Stimulus/Response Sequences

* Parallax scrolling/clicking on the Doctor Octocat
  + Each main div slide transitions from section to section giving a dynamic feel, as the background moves at a different speed to the scroll speed.
* Motion Gif animation, that moves as you scroll, mixed with a moving parallax background
  + gives as sense of user control. Each frame moves as the background moves.
* CSS3 and jQuery Rotating Gallery
  + Allows the user to rotate through the images of the featured cats which there is information on, the stars of the show
* Fruit Machine Game with Customised Modal Box pop up
  + A game of chance. Match three images and you win a coupon code for the site! The system will prompt you if you are lucky, or if you are not lucky this time. The code was based of a classic “one armed bandit” machine, and edited to our applications specifications. A pop up alert notification is included. This informs the user whether they have won or not, using the alertify.js library.

##### CSS3 Tabbed Information

Description

This features on the home page, displaying information on each of the product types available in the application, using css3 tabbed boxes.

Stimulus/Response Sequences

* By clicking the tabs, users can learn all about the different products
  + Each tab has an image
  + Facts
  + Description

##### Gallery using Lightbox

Description

A CSS based lightbox gallery (which is featured next to a more traditional jQuery lightbox gallery) demonstrates images and information about t-shirts that users of the site have submitted to be put on display.

Stimulus/Response Sequences

* Modal-Style pop up
* Information box
* Titles featured on :hover

##### CSS3 Typography

Sticking to the use of web-safe fonts would have limited our choices for aesthetically pleasing typography throughout the site. In order to expand the range of fonts available to our team, we heavily utilised the new typographic features, which were introduced with CSS3. An extensive library of fonts can be found within the assets folder of the application. These were called with the @font-face command.

@font-face {

   font-family: 'BebasNeueRegular';

   src: url('fonts/BebasNeue-webfont.eot');

   src: url('fonts/BebasNeue-webfont.eot?#iefix') format('embedded-opentype'),

        url('fonts/BebasNeue-webfont.woff') format('woff'),

        url('fonts/BebasNeue-webfont.ttf') format('truetype'),

        url('fonts/BebasNeue-webfont.svg#BebasNeueRegular') format('svg');

   font-weight: normal;

   font-style: normal;

}

We also took advantage of the Google Font API, which hosts a range of high quality, open source typefaces.

@import url(http://fonts.googleapis.com/css?family=Arvo);

We further enhanced typography by using the lettering.js jQuery plug in. It allows for dynamic styling of headings, which we talk about in the next section in more detail.

#### 3.1.5 JavaScript Frameworks and Toolkits - Extending JavaScript

##### jQuery or Angular or both?

jQuery was used extensively in the application in order to provide a more dynamic, interactive user experience. The advantages of using jQuery over vanilla JavaScript in the development is that coding time is faster as it is a lightweight, "write less, do more", library. Although performance may suffer as vanilla JavaScript is always faster, for this development the constraints of vanilla JavaScript vs jQuery were negligible.

We had decided to explore angular as well as kinetic.js (for canvas), however due to time constraints of the course, as well as other subject areas, we were forced to focus solely on jQuery. There are several amazing canvas libraries available, however we did not have time to implement them and deliver the application.

jQuery was used in this application to simplify some things that would have been more complicated through JavaScript, such as AJAX calls and DOM manipulation. The purpose of using jQuery is that it makes it much easier to use JavaScript in a site. It reduces the amount of code required to accomplish tasks and wraps them in methods that can be called with a single line of code.

This is demonstrated throughout the site in the following functionality:

##### Fancybox

FancyBox was utilized on the images in the product show pages. This provides users with higher resolution images of the clothing, which was one of our initial requirements for the application. FancyBox displays images, html content and multimedia in a Mac-style "lightbox" that floats on top of the web page. It has a nice drop-shadow effect on the page and is responsive, changing its dimensions automatically to suit the browser window.

products/show.html.erb:

       $(document).ready(function() {

           /\*

            \*  Simple image gallery. Uses default settings

            \*/

           $('.fancybox').fancybox();

           /\*

            \*  Different effects

            \*/

           // Change title type, overlay closing speed

           $(".fancybox-effects-a").fancybox({

               helpers: {

                   title : {

                       type : 'outside'

                   },

                   overlay : {

                       speedOut : 0

                   }

               }

           });

           // Disable opening and closing animations, change title type

           $(".fancybox-effects-b").fancybox({

               openEffect  : 'none',

               closeEffect    : 'none',

               helpers : {

                   title : {

                       type : 'over'

                   }

               }

           });

           // Set custom style, close if clicked, change title type and overlay color

           $(".fancybox-effects-c").fancybox({

               wrapCSS    : 'fancybox-custom',

               closeClick : true,

               openEffect : 'none',

               helpers : {

                   title : {

                       type : 'inside'

                   },

                   overlay : {

                       css : {

                           'background' : 'rgba(238,238,238,0.85)'

                       }

                   }

               }

           });

           // Remove padding, set opening and closing animations, close if clicked and disable overlay

           $(".fancybox-effects-d").fancybox({

               padding: 0,

               openEffect : 'elastic',

               openSpeed  : 150,

               closeEffect : 'elastic',

               closeSpeed  : 150,

               closeClick : true,

               helpers : {

                   overlay : null

               }

           });

           /\*

            \*  Button helper. Disable animations, hide close button, change title type and content

            \*/

           $('.fancybox-buttons').fancybox({

               openEffect  : 'none',

               closeEffect : 'none',

               prevEffect : 'none',

               nextEffect : 'none',

               closeBtn  : false,

               helpers : {

                   title : {

                       type : 'inside'

                   },

                   buttons    : {}

               },

               afterLoad : function() {

                   this.title = 'Image ' + (this.index + 1) + ' of ' + this.group.length + (this.title ? ' - ' + this.title : '');

               }

           });

           /\*

            \*  Thumbnail helper. Disable animations, hide close button, arrows and slide to next gallery item if clicked

            \*/

           $('.fancybox-thumbs').fancybox({

               prevEffect : 'none',

               nextEffect : 'none',

               closeBtn  : false,

               arrows    : false,

               nextClick : true,

               helpers : {

                   thumbs : {

                       width  : 50,

                       height : 50

                   }

               }

           });

           /\*

            \*  Media helper. Group items, disable animations, hide arrows, enable media and button helpers.

           \*/

           $('.fancybox-media')

               .attr('rel', 'media-gallery')

               .fancybox({

                   openEffect : 'none',

                   closeEffect : 'none',

                   prevEffect : 'none',

                   nextEffect : 'none',

                   arrows : false,

                   helpers : {

                       media : {},

                       buttons : {}

                   }

               });

       });

##### Accordion Navigation

The accordion expands and collapses when users click on the navigation labels which behave as containers. These list items contain secondary lists within, which are the respective sub-menus. It is a nice, intuitive design that adds interest to the page and creates interaction between the user and the site.

For the accordion, the jQuery selector targets only specific anchor links that are directly inside the nav container. Anchors that have sibling <ul> elements are recognized as elements that have a navigation to display, so we don’t load those href values when clicked. Then we check to see if the link is open and if so, the other navigational elements are arranged in response. If the current anchor does not have the class .open, we hide menus on the page that are already open and then expand the newest one only, this way only one menu in the accordion can be open at any given time.

nav.js:

$(document).ready(function(){

 $("#nav > li > a").on("click", function(e){

   if($(this).parent().has("ul")) {

     e.preventDefault();

   }

   if(!$(this).hasClass("open")) {

     // hide any open menus and remove all other classes

     $("#nav li ul").slideUp(350);

     $("#nav li a").removeClass("open");

     // open our new menu and add the open class

     $(this).next("ul").slideDown(350);

     $(this).addClass("open");

   }

   else if($(this).hasClass("open")) {

     $(this).removeClass("open");

     $(this).next("ul").slideUp(350);

   }

 });

});

##### Slider Gallery - UI interaction

A CSS3 and jQuery rotating gallery allows the user to rotate through the images of the featured items, this is incorporated into the parallax scrolling splash page. This is an asymmetrical image slider with a twist: when sliding the pictures we slightly rotate them and delay the sliding of each element. The unusual shape of the slider is created by some elements placement and the use of thick borders. We also added an autoplay option on this gallery and it has been incorporated into the Parallax splash page.

##### Store Locator - Google Maps API, GeoLocation & JSON

The store locator shows a map of the world with all of the AriaQuen stores plotted onto it. This feature utilises the Google Maps API, GeoLocation and JSON key:value pairs for the address of the stores.

store\_locator.html.erb:

<style>

     html, body, #map\_canvas {

       height: 500px;

       margin: 0px;

       padding: 0px

     }

   </style>

   <div id="map\_canvas"></div>

   <script type="text/javascript">

$(function() {

 $('#map\_canvas').gmap().bind('init', function() {

  // This URL won't work on your localhost, so you need to change it

// see http://en.wikipedia.org/wiki/Same\_origin\_policy

$.getJSON( '../Scripts/stores.json', function(data) {

$.each( data.markers, function(i, marker) {

$('#map\_canvas').gmap('addMarker', {

'position': new google.maps.LatLng(marker.latitude, marker.longitude),

'bounds': true

}).click(function() {

$('#map\_canvas').gmap('openInfoWindow', { 'content': marker.content }, this);

});

});

});

});

});

       </script>

Stores.json:

{"markers":[ { "latitude":40.64897, "longitude":-73.70277,"title":"NY", "content":"<b>Long Island,</b>  <br> USA <br>  <img src='images/storelocator/tshirt-icon.png' />" },

{ "latitude":53.35326, "longitude":-6.26736, "title":"Dublin", "content":"<b>Dublin,</b> <br> Ireland <br> <img src='images/storelocator/tshirt-icon.png' />" },

{ "latitude":54.00196, "longitude":-6.39988, "title":"Dundalk", "content":"<b>Dundalk,</b> <br> Ireland <br> <img src='images/storelocator/tshirt-icon.png' />" },

{ "latitude":51.89890, "longitude":-8.49518, "title":"Cork", "content":"<b>Cork,</b> <br> Ireland <br> <img src='images/storelocator/tshirt-icon.png' />" },

{ "latitude":19.01413, "longitude":-98.19069, "title":"Puebla", "content":"<b>Puebla,</b> <br> Mexico <br> <img src='images/storelocator/tshirt-icon.png' />" },

{ "latitude":37.77897, "longitude":-122.42683, "title":"Latina Town", "content":"<b>Latina Town,</b> <br> USA <br> <img src='images/storelocator/tshirt-icon.png' />" },

{ "latitude":48.87352, "longitude":2.33047, "title":"Paris", "content":"<b>Paris,</b> <br> France <br> <img src='images/storelocator/tshirt-icon.png' />" },

{ "latitude":53.35326, "longitude":-6.26736, "title":"Dublin", "content":"<b>Dublin,</b> <br> Ireland <br> <img src='images/storelocator/tshirt-icon.png' />" },

{ "latitude":53.35326, "longitude":-6.26736, "title":"Dublin", "content":"<b>Dublin,</b> <br> Ireland <br> <img src='images/storelocator/tshirt-icon.png' />" },

{ "latitude":53.35326, "longitude":-6.26736, "title":"Dublin", "content":"<b>Dublin,</b> <br> Ireland <br> <img src='images/storelocator/tshirt-icon.png' />" },

{ "latitude":53.35326, "longitude":-6.26736, "title":"Dublin", "content":"<b>Dublin,</b> <br> Ireland <br> <img src='images/storelocator/tshirt-icon.png' />" },

{ "latitude":53.35326, "longitude":-6.26736, "title":"Dublin", "content":"<b>Dublin,</b> <br> Ireland <br> <img src='images/storelocator/tshirt-icon.png' />" },

{ "latitude":53.35326, "longitude":-6.26736, "title":"Dublin", "content":"<b>Dublin,</b> <br> Ireland <br> <img src='images/storelocator/tshirt-icon.png' />" },

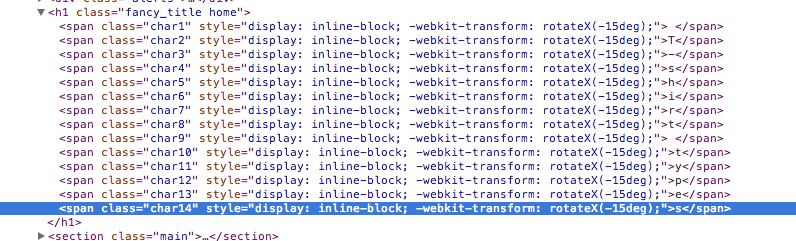
{ "latitude":53.35326, "longitude":-6.26736, "title":"Dublin", "content":"<b>Dublin,</b> <br> Ireland <br> <img src='images/storelocator/tshirt-icon.png' />" }

]}

##### Lettering.js

Lettering.js is a jQuery plugin extension for radical web typography. This was used for the brightly coloured headings that can be found throughout the site. Interactive typography is something our entire team wanted to develop further, given our respective backgrounds in Graphic Design. Some dynamic, interactive functionality was added to the lettering on both the Affiliates page and various Product Categories pages.

Lettering.js manipulates DOM elements by using the .lettering function to place each word or letter, depending on what you wish to do, within a span, with a class of .char# where # is equal to the respective number in the sequence via JavaScipt DOM manipulation.



Ideally, given more time, we would employ some more typographic functionality and behaviours to other pages in the application. More information on typography throughout the site is included in the section on CSS3 Typography, which can be found above.







##### jQuery T-shirt Gallery

Least.js was used for this gallery, which allows users with legacy browsers to benefit from the same functionality that is offered through the CSS3 t-shirt gallery.

It is responsive and invokes the least.js and lazyload.min.js plugins. Lazy Load is used to delay the loading of images in long web pages. Images outside of viewport are not loaded until user scrolls to them. This is opposite of image preloading. Using Lazy Load on long web pages will make the page load faster. In some cases it can also help to reduce server load.

gallery.html.erb:

<script src="js/jquery.min.js"></script>

       <script src="js/jquery.lazyload.min.js"></script>

       <script src="js/least.min.js?03072013"></script>

       <script>

       $(document).ready(function(){

           $('#gallery').least();

           /\* layer fade in \*/

           $('.layer').delay('5000').fadeIn();

           /\* hide layer \*/

           $('.close').on(

               'click',

               function(event) {

                   event.preventDefault();

                   $('.layer').fadeOut();

               }

           );

           $(document).keydown(function(event) {

               if(event.keyCode === 27){ $('.layer').fadeOut(); }

           });

       });

       </script>

       <script>!function(d,s,id){var js,fjs=d.getElementsByTagName(s)[0],p=/^http:/.test(d.location)?'http':'https';if(!d.getElementById(id)){js=d.createElement(s);js.id=id;js.src=p+'://platform.twitter.com/widgets.js';fjs.parentNode.insertBefore(js,fjs);}}(document, 'script', 'twitter-wjs');</script>

       <script type="text/javascript">

         (function() {

           var po = document.createElement('script'); po.type = 'text/javascript'; po.async = true;

           po.src = 'https://apis.google.com/js/plusone.js';

           var s = document.getElementsByTagName('script')[0]; s.parentNode.insertBefore(po, s);

         })();

       </script>

       <script type="text/javascript">

         var \_gaq = \_gaq || [];

         \_gaq.push(['\_setAccount', 'UA-16040332-5']);

         \_gaq.push(['\_trackPageview']);

         (function() {

           var ga = document.createElement('script'); ga.type = 'text/javascript'; ga.async = true;

           ga.src = ('https:' == document.location.protocol ? 'https://' : 'http://') + 'stats.g.doubleclick.net/dc.js';

           var s = document.getElementsByTagName('script')[0]; s.parentNode.insertBefore(ga, s);

         })();

       </script>

##### Social Media Buttons

Twitter Newsfeed API

A Twitter newsfeed is included on the Social Media page of the application. This was easily added with JavaScript, through the use of a Twitter REST API. It fetches the newest tweets from the AriaQuen Twitter account, each time the page is reloaded.

Facebook Follow

Users can follow AriaQuen on Facebook by clicking this button.

Pin it on Pinterest

Users can pin content from the site onto their Pinterest boards.

social media code:

<h1 class="fancy\_title">Social Media</h1>

<a class="twitter-timeline"  href="https://twitter.com/ariaquen" data-widget-id="279957117404844032">Tweets by @ariaquen</a>

       <ul style="list-style: none;">

       <li><a href="https://twitter.com/ariaquen" class="twitter-follow-button" data-show-count="false">Follow @ariaquen</a>

<script>!function(d,s,id){var js,fjs=d.getElementsByTagName(s)[0];if(!d.getElementById(id)){js=d.createElement(s);js.id=id;js.src="//platform.twitter.com/widgets.js";fjs.parentNode.insertBefore(js,fjs);}}(document,"script","twitter-wjs");</script></li>

       <li><div class="fb-follow" data-href="https://www.facebook.com/ariaquen " data-layout="button\_count" data-show-faces="true" data-width="450"></div></li>

       <li><a href="http://pinterest.com/ariaquen/"><img src="http://passets-lt.pinterest.com/images/about/buttons/pinterest-button.png" width="80" height="28" alt="Follow Me on Pinterest" /></a></li>

       </ul>

       </div>

* jQuery and jQuery UI
* Angular
* Backbone
* ember.js
* HTML5 Canvas Toolkits
  + Chart.js
  + Canvas.js
  + Kinetic.js (2d)
  + [Processing.js](http://en.wikipedia.org/wiki/Processing.js)
  + Three.js (3d)

We had planned on using Canvas Libraries, to manipulate the canvas element to allow users to design their own t-shirts, however due to the sheer scope of the project, we did not have time to implement this functionality as a team, as due to time and other commitments, other functionality, which was deemed to be more important got precedence.

##### Pre-compiled HTML, CSS and JavaScript

Haml and Jade

We decided against using HAML or JADE, which are pre-processed code format for HTML. We decided that it would be too much to take on, however we would be interested in using it in the future.

Less and Sass

Less and Sass are pre-processed CSS frameworks. They allow you to add programming to the CSS language, making variables, mixin code and functions available to make CSS more dynamic. You can also nest your Sass, thus making it more akin to how your HTML is written, and paralleling scope a bit more realistically.

We decided to use Sass, as it modularises CSS, and allows us to separate concerns to make code more legible, and easier to update. Sass is also the built in pre-processor for Ruby on Rails, so that made the decision to use it all that more easy. We utilised Bourbon as well, which is a simple and lightweight mixin library for Sass.

CoffeeScript

Like HAML and JADE, CoffeeScript is a pre-processed version of JavaScript, aiming to minimise markup and speed up development by reducing errors. We decided against using CoffeeScript, despite it being built into Rails, like Sass. The learning curve was too high, and we were mainly using JavaScript to manipulate jQuery and other libraries.

# 4. User Interface Design

## 4.1 Requirements – Functional and Non-Functional

Before beginning the development process we looked at existing e-commerce websites that provide an enhanced experience for users. One of our main goals was to bring the t-shirt store to the user’s home, workplace or wherever they choose to browse. Our target audience consists of young professionals in the 25-40 age range who often lead very hectic lives and do not always have time to visit a bricks-and-mortar outlet. Shopping online offers them the opportunity to view and purchase merchandise 24/7 without taking a lot of time out of their day to do so. Products are shipped straight to their homes and are easily returnable if any problems are encountered.

With a virtual shopping experience, users cannot physically try the clothing on or touch the fabrics, so they want to get a feel for the store’s merchandise through the images available. Usability testing revealed that customers like to see many high-resolution images showing products at many different angles. They also need clean, intuitive navigation and knowing that their sensitive information is secure at all times.

The fundamental objectives of the development were as follows:

* To sell goods which include bespoke clothing and accessories.
* To provide an interactive, dynamic user experience.
* Provide effective navigation on the site with uncluttered pages.
* Create meaningful content and interactions.
* Showcase the work of customers who have used our site previously.
* Build and advertise the brand online, increasing awareness of the company.
* Provide various methods of communication between business owners and customers via the blog, forum, enquiry forms and links to social media.
* Provide updates on the creative process and upcoming events.
* Provide a space for logged-in users to personalize their profiles, including profile pictures and personal information that can be updated.
* To provide a successful e-commerce/user experience that customers will want to return to again and again.
* Minimal effort should be required for browsing, searching, finding, comprehending, consuming and interacting.
* To build a site that is learnable, operable, accessible and effective.

## 4.2 Considerations

By testing our JavaScript code, we know our system is always working, though if JavaScript is disabled, only static content will show, and it will lose its rich functionality. We have used simple, comprehensible language, which the user can relate to, by doing so we believe that the user is always in control in the app. Each area of the application is clearly marked in the nav bar, which related to typical standards (header, navigation, footer).

The application aims to be cross-browser compatible. We have done this by including “normalize.css” - a css file which normalizes functions as part of our CSS bundle, as well as the modernizr.js HTML5 shim, which converts HTML5 code to more legacy friendly markup). On the gallery page two types of gallery can be found, one with CSS3 and the other with jQuery. This maintains the intended functionality across both modern and legacy browsers, ensuring that the user will be able to view the range of designs on show.

## 4.3 Responsiveness

Under consideration for responsiveness in the application:

* Skel.js
* Skeleton
* Boilerplate
* Bootstrap
* Fancybox
* Bootstrap
* CSS3 t-shirt gallery

The application uses jQuery functionality to manipulate DOM elements. The shell of the application is wrapped in Skeleton CSS, which offers a grid-based system. In retrospect, we would have preferred to used GRID 960 CSS responsive functionality, or employed the shell of HTML5 Boilerplate, as these allow more customisation. We decided not to use Twitter Bootstrap, as most sites that ulitise Bootstrap all look the same.

Fancybox is used as the gallery lightbox, as it is lightweight, clean crisp and much neater than lightbox.js. It is responsive and changes to fit the browser screen.

The CSS3 t-shirt gallery is also responsive, changing the size of gallery images to fit the browser screen.

As a team we did not have a great deal of experience in creating responsive applications, and it was something we wished to explore in this project. As a result of the in-class tutorials and extensive research done, our knowledge of this area has improved.

# 5. Application Architecture

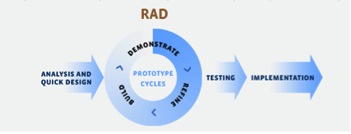
The server side of the application is Ruby on Rails. We chose rails over C#.net as we are most comfortable with Rails and time could be better spent utilising newer front end technologies, rather than learning another server side language. Rails allowed us to build some advanced functionality on the back end, which nicely complimented the use of the front end technologies we employed.

## 5.1 Development and Analysis Methods

This application combines our knowledge of client and server side technologies. The front end of the site has been built with HTML5, CSS3, JavaScript, jQuery, JSON, Ajax and APIs. For the back end we utilized the Rails application framework with Ruby as our programming language, and the Model-View-Controller system architecture. The entire application was built in Mac OS X with the Aptana Interactive Development Environment. RAD and Prototyping methodologies were implemented to ensure the app was developed quickly with the appropriate changes made after each iteration.

## 5.2 Software Process

An agile approach was employed during the development of the app to ensure that the prototype and beta versions had minimal functionality at least. This was then built onto in many iterative steps, as in Rapid Application Development. During the requirements gathering stage we had a broad understanding of the customers’ needs and what we needed to build, with many changes expected along the way. There was also a limited amount of time in which to build the application. RAD involves iterative development and the construction of prototypes, this allows the developers to do some analysis, design and implementation before receiving feedback. Then the cycle repeats. This supports human learning a lot better and enhances the accuracy of the developers’ perception of what clients and users expect. Rapid Application Development is best suited to small-scale projects with a narrow focus, involving few people and a small number of business practices.



## 5.3 Security

As our application is built with rails on the back end, it is sanitised from HTML and SQL injection. Rails includes built-in security modules. For the login and registration functionality we used the Devise gem, which comes with form validation by default. Extra form validation has been added on the client side, in order to sanitise the inputs for the image upload facility. This was done through regular expressions written in vanilla JavaScript, to prevent malicious code injection.

## 5.4 Error Handling

Custom error messages were designed for the application, to deal with 404 “page not found” errors, a variety of server errors and issues with redirection on the site.

## 5.5 Toolkits and Frameworks

Front-end wise our app is powered through Sass with CSS3 features.

* Sassy CSS (.scss) - Used
* Modernizr/html shiv - Used
* CoffeeScript - Not Used
* Angular.js - Not used
* Skel.js - Used
* jQuery and jQuery UI - Used
* Dropzone.JS - Began Implementation
* Bootstrap - Not Used
* Flickr API - Used for Affiliates & Social Media
* Google Maps API - Used for Store Locator
* HTML5 Canvas Toolkits - Not used due to time constraints
  + Chart.js
  + Canvas.js
  + Kinetic.js (2d)
  + Processing.js
  + knockout.js
  + Three.js (3d)

We used Sass, and intended using CoffeeScript in the application, however we decided to stick to vanilla JavaScript notation, as it Coffee Script was syntactically much different and too much of a learning curve. As all CSS is valid Sass, we used a mix of Sass functions, as well as vanilla CSS (as only ⅓ of the team knows Sass well).

Modernizr HTML5 Shim (which includes many shims, shivs, polyfills and fallbacks) caters for legacy browsers who cannot render HTML5 and CSS3 features, such as IE8, and older versions of competent browsers like Firefox and Chrome. By implementing Modernizr, we can cater for people who choose not to update their browsers.

Due to time constraints we could not implement Angular (though we installed the angular Gem for rails). Angular is a popular MVW client side logic framework for creating apps. it has many robust features, however it can be problematic to deploy a rails app with angular to heroku and other platforms without specific customisation, therefore we decided to stick to simple jQuery.

Skel.js and Skeleton were used to implement the responsive UI features.

Dropzone, a jQuery UI plugin was implemented so as to render drag and drop uploads, however due to time constraints this remains unfinished.

jQuery and jQuery UI are a library for multifunctional JavaScript and CSS. It is the defacto industry standard for contemporary JavaScript UI development, and used by many in industry. A problem with reusing others code, is that is sometimes optimised for older versions of jQuery, so it is necessary to rewrite much code to make it compliant with the most up to date CDN repository.

Google maps API, along with HTML5 Geo-location was used to build the storelocator, as discussed previously.

Flickr API was used to pull in affiliate information by searching the persons’ Flickr account for suitable images. The user is encouraged to click on a name, and then the images are loaded with a JSON feed from Flickr via Ajax and nicely displayed in a grid. This feature was also utilised on the Social Media page, which includes a Twitter newsfeed. This was easily added with JavaScript, through the use of a Twitter REST API. It fetches the newest tweets from the AriaQuen Twitter account, each time the page is reloaded.

As mentioned before, we did not have time to implement HTML5 Canvas features, however Kinetic.js looked the most promising for allowing to draw and render images and save to local storage, and later upload to the database.

## 5.6 Data Transfer Strategies

Ajax and JSON encapsulation were used in this application to retrieve data from the server asynchronously (in the background) without interfering with the display and behavior of the existing page. This makes the viewing of information much faster, acting as an intermediary between the front end and server side code. The XMLHttpRequest object enables JavaScript to make HTTP requests to a remote server without the need to reload the page. HTTP requests are made and responses received, entirely in the background and without the user experiencing any interruption.

Switching up the contents of an element without requiring a page refresh adds a “wow” factor to the site, along with added convenience for customers. Furthermore, JSON allows us to store information in an organized, easy-to-access manner. It uses human-readable code to transmit a collection of data consisting of attribute-value pairs.

This is demonstrated in two key features on the site:

### Advanced Product Search

Users can search the database for products in the AriaQuen store, through a specific product search page, which works with Ajax and jQuery to stop the url from changing every time a request is made. As the customer types into the search bar the results show without the need to ever press the submit button. These results can be sorted by name, brand (men’s, women’s and children’s) and new-in.

The product search results automatically update when a user keys anything into the form. For this we use a listener on the submit event, listen for the input field’s keyup event and add a callback so that we can use an Ajax request inside of it. We serialize all the attributes so that it submits all the form variables and it after a response is returned it runs the script. Pagination is also built into the table of products, which works asynchronously too. For this we used the Will\_Paginate gem.

This functionality was all built with the help of RailsCasts tutorials. We focused on creating sortable columns in rails and fetching data from the server with Ajax without requiring the entire page to refresh, which would have disrupted the flow of the application.

application.js:

$(function() {

 $(document).on("click","#products th a, #products .pagination a", function() {

   $.getScript(this.href);

   return false;

 });

 $("#products\_search input").keyup(function() {

  console.log($("#products\_search").attr("action"));

  console.log($("#products\_search").serialize());

   $.get(

    $("#products\_search").attr("action"),

    $("#products\_search").serialize(),

    null,

    "script");

   return false;

 });

});

### Flickr Gallery

AriaQuen’s Affiliates is a feature in the app that pulls in the latest Flickr images from various affiliates of the brand, including other clothing designers, photographers and artists. The user is encouraged to click on a name, and then the images are loaded with a JSON feed from Flickr via Ajax and nicely displayed in a grid.

The Flickr gallery uses jQuery and Ajax to request the latest photos from certain Flickr feeds, along with flags asking for a JSON-formatted response. Users are easily identified by their Flickr ID, which is then passed as an argument called flickrID. A function called loadFlickr() loads the JSON response.

Many social media sites provide RSS feeds, which are easy to import and use on the server-side of an application, but loading them with Ajax is not possible. Due to client-side security, an RSS feed can only be loaded if we request it from the same domain that it’s hosted on. Instead we used a method called JSONP that uses a callback function to send the JSON data back to the site. Each photograph is stored in an array called items, which is accessed in the Ajax call by using feed.items. Then it loops through each item to get the data for each entry. It then displays the latest 16 photographs in thumbnail form, which are floated in a gallery div and clickable in order to view each one at a higher resolution.

affiliates.js:

var people = ([{

"name":"ANNET PATJILA",

"id":"31349962@N08"

},

{

"name":"TAIZ TEES",

"id":"29503902@N07"

},

{

"name":"BROY CLOTHING",

"id":"90975346@N02"

},

{

"name":"PABLO ABRIL SANCHEZ",

"id":"94524619@N08"

},

{

"name":"FREDRIK SWAHN",

"id":"25709629@N06"

}]);

$(document).ready(function() {

for(i=0, l=people.length; i<l; ++i)

{

$('#users').append(

'<li><a href="javascript:loadFlickr(\''

+people[i].id+'\')">'+people[i].name+"</a></li>\n"

);

}

loadFlickr("31349962@N08");

});

function loadFlickr(flickrid)

{

// Display a loading icon in our display element

$('#feed').html('<span><img src="assets/lightbox-ico-loading.gif" /></span>');

// Request the JSON and process it

$.ajax({

type:'GET',

url:"http://api.flickr.com/services/feeds/photos\_public.gne",

data:"id="+flickrid+"&lang=en-us&format=json&jsoncallback=?",

success:function(feed) {

// Create an empty array to store images

var thumbs = [];

// Loop through the items

for(var i=0, l=feed.items.length; i < l && i < 16; ++i)

{

// Manipulate the image to get thumb and medium sizes

var img = feed.items[i].media.m.replace(

/^(.\*?)\_m\.jpg$/,

'<a href="$1.jpg"><img src="$1\_s.jpg" alt="" /></a>'

);

// Add the new element to the array

thumbs.push(img);

}

// Display the thumbnails on the page

$('#feed').html(thumbs.join(''));

// A function to add a lightbox effect

addLB();

},

dataType:'jsonp'

});

}

                $(document).ready(function () {

                    $(".fancy\_title").lettering().children("span").css({ 'display': 'inline-block', '-webkit-transform': 'rotateX(-15deg)' });

                });

## 5.7 Evaluation and Testing

#### 5.7.1 Client Side Testing

#### For our application we developed a JavaScript function that calculates the distance in kilometres between 2 coordinates, taking latitude and longitude values as parameters. This piece of functionality will give us a way to display the nearest shop to the user.

*Calculator.getDistance (lat1,lon1,lat2,lon2)*

Jasmine was used on the Store Locator page in order to test the JavaScript. SpecRunner.html tests for four pieces of functionality:

* Distance Calculator
  + Expectations (Specs)  
    An expectation in Jasmine is an assertion that is either true or false. A spec with all true expectations is a passing spec. A spec with one or more false expectations is a failing spec.We want to make sure that our function is giving accurate results all the time so we need to test the state of the code with the following expectations:
    - It Should Calculate Distance in KM Correctly
    - It Should be able to  handle incorrect data types as parameters

describe("Distance Calculator", function() {

    it("Should Calculate Distance in KM Correctly", function() {

       expect(Calculator.getDistance(59.3293371,13.4877472,59.3225525,13.4619422)).toBe(1.6);

   })

    it("Should be able to handle incorrect data type as Coordinates", function(){

    expect(function() {Calculator.getDistance(59.3293371,"I'm a bad Coordinate",59.3225525,13.4619422) }).toThrow(new Error("Invalid Coordinates"))

   })  
})

* My callout
  + Should hit remote endpoint
  + Should hit a fake server for data

distancecalculator.js:

var Calculator = (function() {

   // Private stuff up here

   var converterValue = 2.2;

   // Public methods here

   return {

       getDistance: function(lat1,lon1,lat2,lon2) {

if ((isNaN(lat1) == true) || (isNaN(lon1) == true) || (isNaN(lat2) == true)  || (isNaN(lon2) == true) ) {

               throw new Error ("Invalid Coordinates");

            }

 var R = 6371; // Radius of the earth in km

 var dLat = deg2rad(lat2-lat1);  // deg2rad below

 var dLon = deg2rad(lon2-lon1);

 var a =

Math.sin(dLat/2) \* Math.sin(dLat/2) +

Math.cos(deg2rad(lat1)) \* Math.cos(deg2rad(lat2)) \*

Math.sin(dLon/2) \* Math.sin(dLon/2)

;

 var c = 2 \* Math.atan2(Math.sqrt(a), Math.sqrt(1-a));

 var d = R \* c; // Distance in km

 return d.toFixed(1)/1;

function deg2rad(deg) {

 return deg \* (Math.PI/180)

}

       }

   }

}())

#### 5.7.2 Speed Testing

Yslow and Speed Tracer are browser extensions that we used to get detailed information on site performance. These analyze web pages and help to identify areas where improvements can be made and performance best practices can be applied. In our Rails application the log files could also be examined to determine which components were taking a long time to execute.

#### 5.7.3 Usability Testing

We looked at UTI methods such as group testing, showing the site to friends and family who had not seen it previously. After doing this usability testing we received a lot of useful feedback on the site.

* Users feel that a secure checkout is very important when making purchases online, they need to know that their sensitive information is secure at all times.
* Users would like to be able to choose between currencies, so the inclusion of a currency-switcher could be a nice addition in the future.
* Users need clean, intuitive navigational elements.
* Usability testing also revealed that customers like to see many high resolution images showing products at many different angles. This part of the application received very positive feedback.

# 6. Summary

We feel a lot of blood sweat and tears have gone into this application, however we also are aware this is merely a stage one application, and a lot more work and functionality would be required to have the desired effects and showcase of our skills, this is merely the prototype taster application. While there is much contemporary and state of the art CSS and JavaScript UI manipulation, there is indeed scope for a lot more, had time permitted, and this is cited below:

What we would have done if we had more time:

* HTML5 Canvas design your own t-shirt
* Transform and position your t-shirt design on different areas of your chosen t-shirt
* jQuery file upload to put on top of Carrierwave
* Currency picker/converter
* Find the stores that are closest to your location
* Angular.js for navigation
* “This site uses cookies” warning
* Different products for each region
* Make it 100% responsive
* Put in a style switcher - Keith, Miguel, Sandy
* Text expander
* Date picker
* Weather app
* Kaleidoscope
* Dress-up game

Features in development that were not completed:

**Dress up Game utilising jQuery UI**

**Description**

The dress up game was developed based on a simple site we saw in pure JavaScript, which we wanted to replicate using simple jQuery. Using jQuery UI’s .draggable(); we were able to target images on the page, and drag them over an image that would be “Dressed up” in a variety of outfits and garments and accessories. The methodology behind this was pure humour, as well as a way of exploring some of the features of jQuery UI. Unfortunately, due to time constraints we were unable to complete development of this feature. Therefore, it is not included in the final application.

**Stimulus/Response Sequences**

Users are stimulated by the variety of dress up outfits available to dress the character up in. Each item is draggable and you can dress up the character in a variety of outfits. Additional functionality to be explored in the future could include saving an image of your dress-up character.

**Image Uploads with jQuery Drag & Drop**

**Description**

On the rails side of the application, the Carrierwave and Image Magick gems were utilized to provide the functionality for the uploading and resizing of images. We explored a variety of methods for adding drag and drop functionality to this on the front end. These included DropzoneJS and jQuery File Upload. Due to time constraints we were unable to complete this part of the project but we would like to in the future.

**Stimulus/Response Sequences**

Users navigate to the AriaQuen Image Upload facility. They can upload an image of their design and position it on the garment they wish to buy. With DropzoneJS, files are draggable from the user’s computer, straight onto the upload area. The next stage of development would be to find a way to position this image over a t-shirt and add the product to the cart. Although the application does include a cart and checkout facility, other advanced design features were not implemented.

**Kaleidoscope Image Gallery**

**Description**

Based on the [Catleidoscope](http://catleidoscope.sergethew.com/) project by Matthew Guy, this gallery is a fun concept where images change with the user’s mouse-movements.

**Stimulus/Response Sequences**

Users can choose from a variety of different images, which change the appearance of the kaleidoscope. As the cursor is moved, the pattern changes on the screen.



**HTML5 Canvas**

**Description**

HTML5 canvas is a powerful too, with several API libraries, replacing largely Flash’s role in browser based game development and implementations. It is a cornerstone of RIA.

**Stimulus/Response Sequences**

Ideally we would have liked to have several canvas features, such as minigames, not disimilar to the fruit machine one arm bandit on the “[coupons](http://guarded-tundra-5012.herokuapp.com/site/discounts)” page, to add user interaction and a greater user interface.

We also aspired, as previously mentioned to implement a drag and drop and design-in-browser feature to allow designers to design their own t-shirts.

**Parallax Homepage**

**Description**

The parallax homepage has many elements and features. However we would like to further develop it, and allow much more functionality within it.

# References

Below are some of the resources used when researching and building this application.

**Websites**

13 Best Responsive CSS Grid Systems for Your Web Designs. 2013. Design Instruct | Jason Gube. [ONLINE] Available at: <http://designinstruct.com/web-design/responsive-css-grid/> [Accessed 10 April 2014].

Lynda.com. 2014. Lynda.com | Lynda.com [ONLINE] Available at: <http://www.lynda.com/> [Accessed April 2014].

Migrating to PostGreSql. 2012. RailsCasts | Ryan Bates. [ONLINE] Available at: <http://railscasts.com/episodes/342-migrating-to-postgresql> [Accessed 1 May 2014].

Sortable Table Columns. 2010. RailsCasts | Ryan Bates. [ONLINE] Available at: <http://railscasts.com/episodes/228-sortable-table-columns> [Accessed March 2014].

Search, Sort, Paginate with Ajax. 2010. RailsCasts | Ryan Bates. [ONLINE] Available at: <http://railscasts.com/episodes/240-search-sort-paginate-with-ajax> [Accessed March 2014].

Carrierwave File Uploads. 2011. RailsCasts | Ryan Bates. [ONLINE] Available at: <http://railscasts.com/episodes/253-carrierwave-file-uploads> [Accessed 25 April 2014].

TeamTreehouse. 2014. TeamTreehouse | TeamTreehouse [ONLINE] Available at: <http://teamtreehouse.com/> [Accessed March 2014].

Learn JavaScript. 2014. Codeacademy  | Codeacademy [ONLINE] Available at: <http://www.codecademy.com/tracks/javascript> [Accessed March 2014].

Stackoverflow. 2014. Stackoverflow | Stackoverflow [ONLINE] Available at: <http://stackoverflow.com/> [Accessed February 2014].

Github. 2014. Github | Github [ONLINE] Available at: [http://www.github.com/](http://github.com)[Accessed January 2014].

Copterlabs. 2014. Copterlabs | Copterlabs [ONLINE] Available at: <http://www.copterlabs.com/> [Accessed April 2014].

Fancybox. 2014. Fancybox | Fancybox [ONLINE] Available at: <http://www.fancybox.net/> [Accessed March 2014].

Vandelay Design. 2014. Vandelay Design | Vandelay Design [ONLINE] Available at: <http://www.vandelaydesign.com/> [Accessed March 2014].

Tympanus. 2014. Tympanus | Tympanus [ONLINE] Available at: <http://www.tympanus.net/>[Accessed February 2014].

Jakob Nielsen. 2006. Quotations from Jakob Nielsen | Dey Alexander Consulting [ONLINE] Available at: <http://www.deyalexander.com.au/resources/quotations/jakob-nielsen.html>  [Accessed January 2014].

IE6 Countdown to Death. 2014. IE6 Countdown to Death | Ryan Parr [ONLINE] Available at: <http://www.ie6death.com/> [Accessed April 2014].

**Books**

Larry Ullman (2012). Modern JavaScript: Develop and Design. United States: Peachpit Press.

Sam Ruby (2013). Agile development with rails - Edition 4. United States: O'Reilly Media.

Chris Pine (2009). Learn to Program in Ruby. United States: Pragmatic Bookshelf.

Ellie Quigley (2010). JavaScript by Example. United States: Prentice Hall Ptr.

Five Dimensions of User Experience (2006). Balancing business and user experience perspectives to create successful e-commerce sites. Human Factors International. Phil H. Goddard / VP Western Region.