National College of Ireland

M. Sc. in Web Technologies

2013/2014

|  |  |
| --- | --- |
| **Student Name** | **Student ID** |
| Patrick Dunlevy | 12116921 |
| Paul Harrison | 00111132 |
| Mark Likeman | 12110141 |
| Niall Ryan | 12114812 |

**BottleRocket**

Project in

Advanced Rich Internet Applications



• Motivation [5%]

• Project Scope:

This is a short description of why your innovation is useful and what it might achieve.

BottleRocket is a Rich Internet Application for fans of music and movies. The application allows users to search for information about musicians and artists, stream videos and music by those artists, search for upcoming music events in their area and find information about new movies that are being released in their locality.

• Area of contribution:

This should describe the general problem area. For example, what is a problem in educational environments and how might they be better developed through the application of a RIA.

Problems?  
Complex Server Logic  
Proprietary Data Solutions  
Desktop-like Application Experience  
SPAs, Web Services, Client-Side Technologies, Cross-Platform

BottleRocket is an entertainment mashup, a concept that was foreign to many only a couple of years ago, but is now extremely common on the web. By taking advantage of massive open datasets, provided by various services through their APIs, and the increasing power of client-side web technologies, BottleRocket offers a unique user experience that is easily implemented across many different platforms.

Web applications have evolved significantly in recent years. Whereas 2-3 years ago an application like BottleRocket would have been easy to create using a framework such as Ruby on Rails, the delivery of the application to user’s web browsers would have been slow, and offered a less positive user experience than the same application could offer if it was traditional native software solution.

BottleRocket takes advantage of the emergence of new technologies and frameworks, in conjunction with various open web services to offer an extremely efficient Single Page Application with all processing and business logic contained on the client to provide users with a fast and efficient user experience. By utilising web technologies, BottleRocket can be used on any device that has access to a web browser and has JavaScript enabled, offering the same user experience, regardless of where the user is; all that matters is that they have a good internet connection and that their device is relatively modern.

• State of the Art Review [30%]

• 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.

Investigation into current practice  
Case Studies (Seevl?)  
Academic/Industrial Research  
Structure, Modularity, Testability, MVC/MVW/MV\*

The explosion in power of mobile devices in recent years, combined with powerful new and emerging web standards and technologies, has drastically altered the landscape of web applications. While early Rich Internet Applications would have developed using technologies such as Microsoft’s Silverlight and Adobe Flash, the popularity of smartphones and tablets, which will not run those technologies, combined with the continual exponential increase in computing power generally, has led to W3C standards HTML5 and CSS3 becoming the de-facto technologies for structuring and styling Rich Internet Applications, with JavaScript (ECMAScript) providing the behaviour. The general scope of web applications has changed in recent years also. Static sites with a couple of HTML files, a CSS file and one script are fast becoming forgotten as web applications become more and more complex, utilising the modern web standards I have spoken about and frameworks and toolsets to support developers and ease the cost and time-taken to develop an application. Recent years have seen a move towards complex web applications using time-tested software engineering design patterns to structure the code and make it more modular and testable, e.g. Ruby On Rails, Django, both of which make use of the MVC pattern for developing a web application, but these solutions are all server-based, with the HTML, CSS and JavaScript that the user sees being rendered and pushed to the client by the application server. Recent years however have seen the emergence of toolsets and frameworks that allow developers to easily build complex and modular applications that live solely on the client-side, and communicate with a server-based API to retrieve data, what we know as a modern Rich Internet Application. A RIA solution that is built using only client-side technologies is a very beneficial thing for a company as it allows for some of the complex business logic of an application to be pushed completely to the client, reducing the load on the server, and makes it easy to create cross-platform applications that will work in any web browser as the client-side apps are built using standard web technologies. While the performance of client-side RIAs may have been an issue in recent years, the improvements offered by HTML5 and CSS3 and ECMAScript 5 and 6 make Rich Internet Applications built using these technologies as performant as native solutions while also offering an almost identical user experience.

The growth in popularity of HTML5, CSS3 and JavaScript-based Rich Internet Applications has led to some interesting software engineering problems being created for web developers. As mentioned above, the problem of how to structure a traditional website or web application was almost trivial; early sites and applications contained only minimal static content, and Flash or Silverlight-based solutions had Adobe and Microsoft to rely on for best practices and to understand how to architect a system. However, the problem of how to structure a modern client-side web application has only just become to be understood. While it became usual for web developers to keep all of their JavaScript code in one script file, this is obviously bad practice in a modern web application, where the script file could end up being many thousands of lines long. With the emergence of web development as a software engineering discipline, developers have begun to expect that their code be modular and testable, like a traditional desktop application. The best way to ensure modularity and, therefore, testability is to enforce a commonly-used architecture or design pattern on the system being developed, and so a number of client-side frameworks have emerged in recent months and years to try to help web developers solve these problems which were not traditionally associated with web development.

For this project, Team Frankensteer identified three JavaScript frameworks and toolsets which could be used to help structure and develop a Rich Internet Application to fulfil the criteria for this project; BackboneJS, EmberJS and AngularJS. Each of these tools offer similar helper methods to ease the cost and time-taken to develop a Rich Internet Application, and each enforces or allows for easy enforcement of a design pattern to help define the application structure. It is commonly believed that each of the three is an MVC (Model View Controller) framework but that is not necessarily the case.

BackboneJS is a JavaScript library that provides functionality for interacting with a server / API to pull data into a Rich Internet Application, while also giving that application structure by allowing developers to implement the MVC pattern in their application, using Backbone’s Models, Collections and Views. So, Backbone is not necessarily a framework but a collection of library functions to allow developers to implement functionality, similar in ways to jQuery. While jQuery is a library that allows developers to easily traverse and manipulate the DOM, Backbone is a library for interacting with an API and structuring an application.  
CASE STUDY?

EmberJS is a client-side MVC framework for creating “ambitious” web applications. Ember features View templating, which implements the Handlebars library, enforces modularity by allowing developers to encapsulate files as Ember Components and to abstract functions into Ember classes/models, helper methods for easily retrieving data from a server and helper methods for easily creating complex Routing. Ember is an opinionated client-side framework that emphasises convention over configuration, allowing developers to create complex applications as long as they do things the “Ember Way”. Ember’s opinionated nature sets it apart from Backbone as a framework instead of a library and is growing in popularity among certain sections of the RIA community.  
CASE STUDY

AngularJS is a “toolset for building the framework most suited to your application development”. (CITATION NEEDED) This is an important distinction to draw as it sets Angular apart from the other tools outlined above. Backbone can be considered a library as it extends the functionality with a set of wrappers for JavaScript functions and helper methods to extend the functionality of standard JavaScript. Ember is an opinionated MVC framework which offers developers a number of best-practices for developing a complex client-side MVC application and emphasises convention over configuration. Angular, however, is a toolset that allows developers to structure an application in any way they want, an MVW or MV\* toolset that is completely extensible meaning developers can pick and choose what modules they would like to use to help develop their application. Angular is opinionated but not to the extent that it forces developers to use a certain pattern. Theoretically, Angular can be used to create a completely custom application framework for developing specific styles of Rich Internet Applications. A great example of this is the Ionic framework which integrates AngularJS with PhoneGap for creating mobile applications using JavaScript that can be packaged for Android and iOS devices and sold on the respective platform’s App Stores. While Angular can still be considered to be in its infancy, it has achieved great popularity in recent months and is being used to create many complex Rich Internet Applications in similar domains to the proposed BottleRocket application. An example of this is the Seevl music-discovery platform which uses AngularJS as the front-end toolset for its user-facing Rich Internet Application product, seevl.fm. Originally a standard HTML, CSS and plain JavaScript application, seevl.fm was migrated to AngularJS in June 2013, and this migration led to enhanced performance and a smaller (by almost 60%, a huge saving), more modular and stable code-base that was easily testable. (https://twitter.com/terraces/status/447367876681678849)

After careful investigation, Team Frankensteer decided to use AngularJS to create the BottleRocket application, due in part to the familiarity certain team members already had with the toolset, but also due to the large community and ecosystem of modules that exist for helping to create Rich Internet Applications using the toolset. The Angular-Seed application structure template was also a big factor in deciding to use Angular as it helps to enforce an MV\* pattern and best-practices which would allow for easily separating the different aspects of the application code into modules which ensures that each layer of the application is testable, independent of other modules.  
Critical Analysis of selected approach  
Informed decision about innovation undertaken and technologies

• User Interface Design [20%]

• This section should outline how you arrived at the design of your UI. You should include a detailed analysis of how the controls in your application deliver a successful UI experience.

Mark and Paul

• Architecture [25%]

• Application Architecture:

Outline the solution architecture for your application. You should justify the approach you took, and how it supports the project scope. This means mapping back to the state of the art review when describing the features and functionality of your system.

Solution Architecture => Angular, angular-seed  
 map back to state of the art review

As we decided to develop the BottleRocket application using the AngularJS toolset, the application architecture was very easy to decide upon. AngularJS allows developers to create an extremely modular application structure and provides a template application for developers to build upon with many best-practices and examples already in place, known as Angular-Seed, made available from the AngularJS website by the Angular Core Development Team. Angular-Seed provides a simple and efficient way to enforce the MVC pattern in a client-side application using AngularJS, allowing for efficient modularity and testability.

Angular-Seed creates a scaffolded application with separate Configuration, Service, Controller, Directive, Filter and Presentation modules. The app configuration file, app.js in BottleRocket, is where each of the modules that make up the application are injected as dependencies in the core application module. The Configuration module also contains routing logic for the application.

The Service module is where most of the business logic of the application lives. In the case of BottleRocket, most of the services are AJAX requests using Angular’s $http service. The service module can be seen as similar to the Model folder in a Ruby on Rails application. There are four different types of Service objects that can be created: Constants, Values, Services and Factories.

Directives are unique to Angular and allow encapsulation of HTML and CSS element templates with JavaScript functionality, allowing users to create their own custom elements and call them in the view code e.g <custom-element></custom-element>. Directives are one of Angular’s “wow” features but are not used extensively in the BottleRocket application due to limited development time and also due to the nature of the application; the benefits of Directives are encapsulation for re-use and easy testability but there aren’t many objects re-used extensively in the application and refactoring the code into directives would have added un-needed complexity.

The Controller module is where the business logic from our services is called into play. As in most MVC applications, our controller module can be seen as the glue that holds the application together, calling the data from our services/models and passing this data to our presentation layer, using Angular helper methods. Some of the controllers also contain business logic due to the implementation of SDKs which fall outside of Angular’s $digest cycle (basically, as the SDKs are not Angular modules, Angular is not aware of them), but careful refactoring of some of the SDK code into services and directives would solve this problem. However, due to time constraints, some of this code still lives in the controllers.

• Security:

Evaluate industry standard error handling, and outline how you integrated these approaches within your application.

JavaScript Error Handling?

• Toolkits and Frameworks:

• Discuss the toolkits and frameworks used and the justification for using them.

Angular, angular-seed, Zurb Foundation  
Justification

• Data Transfer Strategies:

• This section looks at the access and transporting of data to be consumed/created by the RIA.

Angular’s $http service, Soundcloud SDK, Seevl, YouTube SDK, BandsInTown, SongKick, Rotten Tomatoes

• Evaluation and Testing:

You must include a short description of how your application was evaluated for its audience. Critically analyse the testing methodology employed, as well as any debugging techniques you used in building the application.

Audience Evaluation => Usability -> Mark and Paul  
Testing Methodology => angular-seed, Jasmine, Karma

• Summary [10%]

• This is a short section that includes a brief summary of what was achieved so far. Evaluate the approach you took, the tools you used, and the implementation of your applications. You should describe what changes you would make or future work that would benefit your application.

Evaluate: Approach, Tools, Implementation  
Development Expansion Plan  
Future of Angular + RIAs?  
PhoneGap, Ionic

• References [10%]

• Please refer to the Harvard guidebook. Ensure to correctly reference all resources using the Harvard style of referencing.