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  
Critical Analysis of selected approach  
Informed decision about innovation undertaken and technologies

Structure, Modularity, Testability, MVC/MVW/MV\*

• 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?

• References [10%]

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