Waqas Aslam Wa40

Alex Austin Aa745

James Grant Jhg7

Fred Barnes

CO600: Language Selection

14 March 2016

Language Selection and Environment Automation

Front End

Team refers to:

Waqas Aslam Alex Austin James Grant Frederick Harrington Harry Jones

Selecting our frontend programming language took a bit of careful deliberation. As a team, we all had some varying levels of expertise in a web development environment.

Several programming languages for the front-end were proposed, some of which included:

- Java Applets & Spring MVC

- Ruby on Rails

- Variety of JavaScript Libraries

- Python Web Frameworks

We began by viewing the application from an almost ubiquitous perspective, planning and discussing the organisation of the application, while drilling towards some of the finer elements of what is now known to be Anexd.

It had transpired that everyone sought their personal preference into how the front end should be developed. Through this, we discovered that the team had possessed some form of applied knowledge in JavaScript.

Based on our expertise and the project specification, JavaScript had been identified as a viable solution for what we needed to accomplish. From an academic judgement, we wanted to assure the complexity of the implementation was enough to satisfy our personal learning, while continuously supporting a modern and contemporary interface.

After some careful research, the decision led us to use Angular.Js as our preferred method for developing our frontend.

Additionally, Alex Austin and Harry Jones hold extensive knowledge of the Angular.Js framework and were confident with their understanding of the implementation mentioned above.

**Some Heading**

After carrying out a review of Angular.Js, we found a variety of strengths surrounding the Angular.Js framework and took the confinements of the language into consideration. Angular is a clever way of producing Single Page Applications (SPAs). The SPA implementation gave us the freedom to develop the HTML page once and the rest of the functionality and interaction written in JavaScript. The rest managed through Angular.Js.

Naturally we noticed that Angular is an open source application, sustained by corporations such as Google and a community of developers.

We noted that Angular provided some favourable features which would significantly improve functionality in a clean, streamlined and maintainable manner.

Some additional capabilities which we identified as beneficial towards our project are as follows:

* Creating a single page application between the model and view controllers.
* Angular code is highly testable.
* Reusable components
* A write less, get more functionality design
* Styling is simplified
* Views are in HTML, the business logic written in JavaScript.
* HTML binding capabilities for a responsive experience

To further aid us in the selection process we looked at some constraints which could cause limitations or provide focus in some areas more than others. Some things we found were:

* Since Angular is a Single Page Application, which could be a disadvantage. If a user has Javascript disabled, the user will only see a basic page and potentially lose out on much of the functionality.
* Since Angular.Js is JavaScript, we identified that we needed to focus on some of the security side of the application such as authentication and authorisation

We explored package managers to complement our frontend development environment.

Initially, Gulp was our automation tool of choice as it was quick in comparison to some of the tools we explored. However, due to some technical issues and complications, we decided that it was not working as well as we had originally hoped.

Both Bower and Npm were an alternative to Gulp. We found that Bower could handle the frontend and Npm dealing with the back-end dependencies. We decided that both these packages would help automatically manage dependencies, keeping all frontend components and developers on the same page respectively.