Many Voices Publishing Platform

Technology Review

D. Kevin McGrath & Dr. Kirsten Winters - CS461 Fall 2016

Commix

Steven Powers, Josh Matteson, Evan Tschuy

Abstract

The Many Voices Publishing Platform uses a variety of technologies to handle different aspects of the project, from the user interface to the backend database operations. These technologies enable to the Many Voices Publishing Platform to succeed in delivering a working platform for textbook collaboration.

CONTENTS

1 TECHNOLOGY REVIEW

1.1 Introduction

The Many Voices Publishing Platform is being developed for the purpose of fixing the problems currently associated with the textbook market. We will accomplish this by giving the MVP Platform an easy to use interface, a search bar with a built in results pane, source control, and many other features. Authorship is divided by subsection header.

1.2 Steven

1.2.1 User Interface

Option 1 - React [1]

React is a JavaScript rendering engine that is developed by Facebook. Originally used with Instagram, React is often paired with Redux for added functionality. React is a popular JavaScript library meant for building user interfaces that is component based.

Option 2 - Aurelia [2]

Aurelia is a newer JavaScript client framework for mobile, desktop, and the web, by using simplistic integration.

Option 3 - Ember [3]

Ember uses web components and templates to increase productivity.

Option 4 - Angular2 [4]

Angular2 is a project started by Google for their internal Green Tea project. Angular2 is a widely documented JavaScript cross-platform library that is used to create native mobile and desktop web applications.

Goals

The use of this technology will aid in the development of the user interface. Having a beautiful and scalable user interface will help users interact with the platform more easily, on whatever device they choose to use it on.

Evaluation Criteria

The options are evaluated on

- Ease of Use
- File Size
- Features
- Performance
- Standards Compliance
- Non-Compliance
- Release
- License

Option Comparison

[5]	React	Aurelia	Ember	Angular 2
Ease of Use	Substantial setup required for working system, lots of documentation and tutorials.	Simple setup using NPM and installation	Simple setup using NPM and installation	Substantial setup required for working system, lots of documentation and tutorials.
File Size	156kb to ???kb, due to added frameworks	323kb	435kb	1023kb
Features	View rendering engine with plugin frameworks	Router, Animation, HTTP Client	Router, HTTP Client	Router, HTTP Client
Performance (Paints per Second)	45-50	90-150 (Higher end with additional plugins)	60-100	80-130 (Higher end with additional plugins)
Standards Compliance	ES 2015	HTML, ES 2016, Web Components	HTML, ES 2015	ES 2016
Non-Compliance	JSX	N/A	N/A	NG2 Markup, Dart
Release	15.x	Beta	2.x	Release Candidate
License	BSD	MIT	MIT	MIT

Discussion

All of the chosen options have their pros and cons for our web application. All of them however would be a learning and research experience. Angular2 and React have the benefit of being created by large software companies, Google and Facebook respectively. This means that there will be large adoption and documentation / tutorials available. Aurelia and Ember seem to be easier to implement however, they are much newer products and they have a smaller adoption population. This could prove troublesome if we run into problems. If our implementation ends up being a fork of Ward Cunningham's Federated Wiki, then this decision will be null most likely.

Selection

Initially we were set on using Angular2 as part of the team has experience using this JavaScript library, before meeting with our client. Angular2 has a wide adoption and is used by Google for internal projects so the longevity of the framework is expected to last. With this in mind, we plan to use Angular2 if we need to use a JavaScript framework for our user interface.

1.2.2 Documentation

Option 1 - Microsoft Word [6]

Microsoft Word is a fairly powerful and widespread software that is often used for documentation outside of higher education and the technology industry.

Option 2 - Dozuki [7]

Integrates revisions, approval, and documentation into a single system that allows for efficient workflows for creating customizable and centrally managed documentation.

Option 3 - LaTeX [8]

LaTeX, hereby known as Latex, is a document preparation system often used in the technical and scientific fields for documentation, and is the de facto standard for communication and publication of scientific documents [8].

Goals

An efficient documentation solution that is able to provide the ability to create beautiful and useful documentation of the platform. This documentation will assist the end user with operation of the various parts of the platform.

Evaluation Criteria

The options are evaluated on

- Ease of Use
- File Size
- Features
- Performance
- Cost

Many Voices Publishing Platform

Option Comparison

	Microsoft Word	Dozuki	LaTeX
Ease of Use	Microsoft Word is easy to use for most cases, but can be difficult to use when attempting certain formatting requests, such as creating a table of contents.	Dozuki is a newer documentation solution that spawned from iFixit, Dozuki prides itself on being easy to learn and use, while providing centralized	Latex is able to take a lot of the tediousness one would find in Microsoft Word, allowing the user to focus on the document contents instead of appearance. "Steep learning curve, but not very high" [9]
File Size	Small to Large, depending on embedded images or graphics	Unknown	Often very small due to only text
Features	Moderately customizable, and can be 'pretty'	Highly customizable with CSS and HTML	Highly customizable and 'pretty'
Performance	Can become bogged down by long documents	Web interface limitation, exact performance unknown	Easily handles complex and length documents
Cost	$149 for Office Home & Student 2016	$299 / month for 10 users, $10 Additional Per User	Free

Many Voices Publishing Platform

Discussion

While we are highly experienced using Microsoft Word, the documents become bloated, and sometimes the software is prone to crashes when dealing with very large documents. Dozuki seems like a very nice documentation solution, that is used by Intel and other technology companies. Having a centralized management system allows management to verify everyone is looking at the most updated documentation. This would be a very nice solution, though the cost is very prohibitive, at over $3500 / year. Latex is free, and is able to create highly structured documentation that is also easy to style. Latex is used throughout the science field for ease of use.

Selection

Latex seems to be the best solution of the three, due to current experience using the software, as well as ease of use when creating large documents with multiple sections and headings.

1.2.3 Inclusive Design

1.3 Josh

1.3.1 Testing

Option 1 - Mocha

Mocha is a JavaScript testing framework, loaded with features. It runs on Node.js and also in the browser, making asynchronous testing simple and easy to use. Mocha tests run serially, allowing for flexible and accurate reporting, while mapping uncaught exceptions to the correct test cases. [10]

Option 2 - QUnit

QUnit is a powerful, easy-to-use JavaScript unit testing framework. It's used by the jQuery, jQuery UI and jQuery Mobile projects and is capable of testing any generic JavaScript code. [11]

Option 3 - Jasmine

Jasmine is a behavior-driven development framework for testing JavaScript code. It does not depend on any other JavaScript frameworks. It does not require a DOM. And it has a clean, obvious syntax so that you can easily write tests. [12]

Goals

Using this technology will aid in proper functionality and minimize errors. Without properly testing code, a number of problems can occur that can disrupt and slow down progress in a team. In extreme cases, not properly testing could lead to failure of the application.

Evaluation Criteria

The options are evaluated on

- Ease of Use
- File Size
- Features
- Performance
- Standards Compliance
- Non-Compliance
- Release
- License

Many Voices Publishing Platform

Option Comparison

	Mocha	QUnit	Jasmine

Discussion

It is well known that an application that ensures proper functionality supersedes other applications and, more importantly, competitors. Quality assurance is acknowledged as highly distinguished, and therefore, an attribute deserving of notable consideration. Having this in mind, we will be considering three different testing frameworks: Mocha, QUnit, and Jasmine.

Mocha (more info to come)

QUnit (more info to come)

The Jasmine testing framework is the most common of the three, and considerably so. (more info to come)

Selection

Mocha?

1.3.2 User Authentication

1.3.3 Database

1.4 Evan

1.4.1 Server Back-end

Option 1 - NodeJS

NodeJS is a modern web back-end framework developed by the Node Foundation, primarily led by Joyent. By using JavaScript its language of choice, Node allows developers to use the language's unique concurrency paradigms to quickly develop scalable applications.

Option 2 - Django

The Django framework is a massive web framework developed in Python that comes "batteries included". The Framework includes everything from geo-libraries to support for four different kinds of databases, meaning a large initial learning curve but a large payoff.

Option 3 - Flask

Flask is a micro-framework. It comes with the bare minimum needed to do HTTP handling, leaving what other frameworks come with to an array of choices from third party developers. This means the core framework is quick to learn, but can quickly leave a developer feeling constrained.

Option 4 - Ruby on Rails

Ruby on Rails is the old standard of web frameworks. It was the original batteries included framework, and has over the years been known for its ease of use. However, the framework is quite old and shows some signs of age, using sometimes outdated paradigms and generally being less friendly to beginners than more modern frameworks.

1.4.2 Text Formatting

Option 1 - Markdown

Markdown is a highly lightweight markup language that allows easy, human-readable markup of text to include headings, bold/italic/underline/etc, bullets, and numbered lists. The original markdown does not support things like images, videos, or links; Markdown has various "flavors", or implementations, that sometimes allow for such things.

Option 2 - Restructured Text

Restructured Text is a markup language written in Python for writing documentation, simple websites, etc. It allows for highly varied but still restricted markup; it allows for image embeds, fancy linking, titles, etc. It does not allow users to embed arbitrary elements.

Option 3 - Raw HTML

Storing simply raw HTML allows the greatest flexibility, as it is literally the same elements rendered in browser. Raw HTML allows for things like scripting, video embeds, etc., and as such must be filtered to a restricted subset to be suitable for use in a public-facing scenario.

1.4.3 Password Storage

Option 1 - Bcrypt

Bcrypt is a password hashing function that takes a very large amount of time to crack an individual password – it is designed to be slow. This means a hacker cannot simply crack a database worth of passwords in one sitting, as with older hashes like MD5.

Option 2 - Scrypt

Scrypt is designed to take up large amounts of time, and large amounts of RAM, when hashing. This ensures that a hacker cannot simply buy a powerful CPU and crack passwords with pure power. However, scrypt, being designed more-so for computer hard disk passwords, can take multiple seconds and hundreds of megabytes of RAM to process.

Option 3 - pbkdf2

PBKDF2 is a function that repeatedly hashes a password using the HMAC, or ”keyed-hash message authentication code”, function. For a CPU, cracking a large number of passwords using pbkdf2 is difficult, as it takes a large amount of time to crack an individual password. Using a GPU, however, a large number of hashes can be run in parallel, making it quick to crack with high end hardware.

Option 4 - raw storage

text

1.5 Conclusion

The Many Voices Publishing Platform is a combination of User Interfaces, Documentation, User Centered Design, Testing, User Authentication, Databases, Server Back-end, Text Formatting, Password Storage, and the users themselves. Determining the technologies behind these parts and pieces is a difficult task to accomplish, as many choices can satisfy the requirements of the project. Finding the best solution however is the goal of this document, to provide a clear path forward for the platform as a whole.

REFERENCES

[1] Facebook, "A javascript library for building user interfaces - react," https://facebook.github.io/react/index.html.

[2] Aurelia, "Aurelia," http://aurelia.io/.

[3] Ember, "A framework for creating ambitious web applications," http://emberjs.com/.

[4] Google, "Our framework," http://angular.io/.

[5] R. Eisenberg, "Choosing a javascript framework," https://www.youtube.com/watch?v=6I_GwgoGm1w.

[6] Microsoft, "Microsoft word — document and word processing software," https://products.office.com/en-us/word.

[7] Dozuki, "Visual documentation for a paperless future," http://www.dozuki.com/.

[8] T. L. Project, "Latex - a document preparation system," http://www.latex-project.org/.

[9] D. K. McGrath, "Latex learning curve."

[10] MochaJs, "Mocha, simple, flexible, fun," https://mochajs.org/.

[11] QUnit, "Qunit: A javascript unit testing framework," https://qunitjs.com/.

[12] Jasmine, "Jasmine," https://jasmine.github.io/2.0/introduction.html.