CS CAPSTONE FINAL REPORT

JUNE 13, 2017

MANY VOICES PLATFORM

PREPARED FOR

OREGON STATE UNIVERSITY

CARLOS JENSEN

PREPARED BY

GROUP 61

REMIX

JOSH MATTESON
STEVEN POWERS
EVAN TSCHUY

Abstract

The culmination of documents created during the development of the Many Voices Publishing Platform by Team Remix; Josh Matteson, Steven Powers, and Evan Tschuy for Dr. Carlos Jensen. Details of the implementation, technologies, and requirements are included in this document.

CONTENTS

The Many Voices Publishing Platform was requested by Dr. Carlos Jensen, who is now the Associate Dean of Undergraduate Programs and Associate Professor at Oregon State University. This project was requested because textbook creation is very difficult, and costly, both in time and money that the author has to spend working on their writing. The importance of this project is that having a platform that enables professors to create the textbooks they want by combining and modifying materials already in existence, or by creating their own that will be added to the available materials for all users. The client for this project was also the person that requested the project, Dr. Carlos Jensen. The development team consists of Josh Matteson, Steven Powers and Evan Tschuy. During development of the platform, Josh Matteson focused on frontend development, Steven Powers focused on documentation, management, and frontend development, and Evan Tschuy focused on the backend development of the platform, but also stepped in to work on other aspects of the project when available. Dr. Jensen remained relatively hands off during development, only stepping in to provide feedback about questions brought up through development.

MANY VOICES PUBLISHING PLATFORM

SYSTEMS AND SOFTWARE REQUIREMENTS SPECIFICATION (SSRS)

Version 0.5

October 27, 2016

Prepared for:

Dr. Carlos Jensen, D. Kevin McGrath, and Dr. Kirsten Winters

CS461 Senior Capstone

Prepared by:

Commix

Josh Matteson, Steven Powers, and Evan Tschuy

Oregon State University

Corvallis, OR 97330

Abstract

An overview of the requirements for the Many Voices Publishing Platform. Introduces the Many Voices Publishing Platform, describes in detail the purpose, product functions, user characteristics, assumptions and dependencies, and system level (non-functional) requirements and lists specific requirements that are outside of the platform, system features, and an overview of the development life cycle.

MANY VOICES PUBLISHING PLATFORM SSRS

TABLE OF CONTENTS

CONTENTS

1 INTRODUCTION

1.1 IDENTIFICATION

The software system being considered for development is referred to as the Many Voices Publishing Platform. The customer providing specifications for the system is Dr. Carlos Jensen that is reachable at his email jensenca@engr.orst.edu The ultimate customer, or end-user, of the system will be any user that wants an easier way to combine materials into a book. This is a new project effort, so the version under development is version 0.5 release 0. As the system may be based off of Ward Cunningham's Federated Wiki, the project will have a substantial head start, signified by the 0.5 version number [1].

1.2 PURPOSE

The purpose of the system under development is to provide a collaborative authoring platform that allows for users to combine various materials into a book. While the system will be used by professors and instructors in majority, this document is intended to be read and understood by software designers and coders. The document will also be vetted or approved by Dr. Carlos Jensen, D. Kevin McGrath, and Dr. Kirsten Winters.

1.3 SCOPE

The project is sponsored by Dr. Carlos Jensen, being developed by Josh Matteson, Steven Powers, and Evan Tschuy for completion of CS461 Senior capstone project class. The product is to be operated via a website for wide availability and use.

1.4 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS

Term or Acronym	Definition
Alpha Test	Limited release(s) to selected, outside testers (Friends and Family)
Beta Test	Limited release(s) to cooperating customers wanting early access to developing systems (Professors and other educators)
Final Test	Release of full functionality to customer for approval
SDD	Software Design Document
SSRS	System and Software Requirements Specification
PDF	Portable Document Format, that is able to combine text, graphics, and other information into a single document

PCI	Payment Card Industry, is a proprietary information security standard for credit cards in an effort to reduce credit card fraud
Federated	Individual parts that stand as an individual but can be combined into a single unit
Wiki	A collaborative content editing platform
Scrap	A section of a textbook, which can contain formatted text (markdown or latex), and media
Section	An ordered collection of Scraps belonging to a chapter
Chapter	An ordered collection of Sections and Scraps
Media	A standalone image, figure, or video. Can be embedded in a Scrap
Web Application	An interactive program that can be accessed and is based through a web server instead of being stored on a user's desktop
Source Control	An element of software design management, version control, and is the management of changes to documents, large web sites, computer programs, and other collections of data
User Interface (UI)	The means by which the user and a computer system interact, in particular the use of input devices and software

1.5 REFERENCES

[1] C. Jeffery, Systems and Software Requirements Specification (SSRS) Template, 2nd ed. Moscow: University of Idaho, 2016, pp. 1-22.

1.6 OVERVIEW AND RESTRICTIONS

This document is for limited release only to Oregon State University professors and associated staff of D. Kevin McGrath, Dr. Kirsten Winters, and Jon Dodge, personnel working on the project, and the client, Dr. Carlos Jensen.

2 OVERALL DESCRIPTION

2.1 PRODUCT PERSPECTIVE

The main differentiator from traditional textbook publication methods are the products' micro-writing abilities. A traditional textbook is written by one expert, using a defined curriculum that fits the needs of the author. The curriculum is a generic version of a multitude of classes that is somewhat useful for many professors, rather than very useful for few professors. Instead, our solution will give the tools necessary for individual professors to create the textbook they need to match the curriculum desired. This allows for textbooks that more closely track the intricacies of how classes are taught by different individuals.

2.2 PRODUCT FUNCTIONS

The product will be a document construction service. The most basic usage will be the creation of a textbook out of pre-existing chapters, by searching and selecting from a list. Alternatively, chapters can created using pre-existing sections and media. A chapter or section can be created using a simple input interface that accepts text and LaTeX markup, along with the upload of media like image or perhaps video content.

2.3 USER CHARACTERISTICS

The expected user of the service will be a professor. Professors, in general, have decent means technical skills. Thus, the interface of the product will be highly graphically driven, using modern interface paradigms as defined by industry leaders, such as Google's Material Design [2]. The interface will allow users to not think about the technical specificities of the product, but more-so on the content they are using, the authors of the content, and how they are using it.

2.4 CONSTRAINTS

A regular web browser will be the means of access for users. For the backend server, the main requirement is availability - as a web application, it has no safety- or life-critical components. A high (99.99%) availability should be provided, but otherwise availability, safety, and specific hardware are not main concerns.

The software can, as a web application, be written in a high-level language, with minimal inter-application communication. In the case that the software is built on top of other applications, such as Git, inter-process communication should be handled in a generic way that provides a high-security barrier, preventing malicious users from accessing other, non-specified applications.

2.5 ASSUMPTIONS AND DEPENDENCIES

The product will be designed to run on a generic Linux-with-web server environment. Ward Cunningham's Federated Wiki, a similar project being drawn upon for inspiration, can run on any system with Ruby and CouchDB (a popular NoSQL database) [1]. With the product being either partially or wholly based on the Federated Wiki, similar system requirements are anticipated [1].

2.6 SYSTEM LEVEL (NON-FUNCTIONAL) REQUIREMENTS

2.6.1 Site Dependencies

As a web application, the product has no hard requirements other than high-speed network access and an instance of the CouchDB database management system.

- Hardware: standard virtual machine(s) running on cloud provider (ex: Amazon EC2 t2.medium)
- Possible Database: CouchDB 2.0

2.6.2 Safety, Security and Privacy Requirements

Private information shall be held to any relevant security standards. For instance, any credit card information processed by the product in the textbook creation process will be held to PCI compliance standards. Passwords shall be stored in a securely hashed, non-reversible method, such that infiltration of the product by an unauthorized third party will not be able to result in the theft of user passwords or other private information.

2.6.3 Performance Requirements

A standard user load shall be defined as up to 100 simultaneous users. User information will include things such as text documents, media uploads, and user account data.

Under a standard load:

- 95% of save operations (excepting media upload time) shall be responsive in nature.
- 95% of initial page loads shall shall be responsive in nature.
- 95% of user account operations (login, creation, deletion, etc) shall be responsive in nature.

2.6.4 System and Software Quality

The software shall maintain 99.99% accessibility from the major modern web browsers, including Mozilla FireFox, Google Chrome, and Microsoft Edge. As a non-safety-critical product, reliability shall be attempted but not promised to end users. Testing shall be handled through a suite of automated unit and integration testing, as well as through manual checking of newly written and critical features.

2.6.5 *Packaging and Delivery Requirements*

The executable system and all associated documentation (i.e., SSRS, code listing, test plan (data and results), and user manual) will be delivered to the customer on CD's and/or via email, as specified by the customer at time of delivery. Although document "drops" will occur throughout the system development process, the final, edited version of the above documents will accompany the final, accepted version of the executable system.

2.6.6 *Personnel-related requirements*

The system under development has no special personnel-related characteristics.

2.6.7 *Training-related requirements*

No training materials or expectations will be tied to this project other than the limited help screens built into the software and the accompanying user manual.

2.6.8 *Logistics-Related Requirements*

The software shall run on a standard Amazon Web Services backend instance with another standard instance serving as the database.

2.6.9 *Other Requirements*

No other system requirements are anticipated.

2.6.10 *Precedence and Criticality of Requirements*

The main critical requirements of the system are those relating to payment processing secrecy.

3 SPECIFIC REQUIREMENTS

3.1 EXTERNAL INTERFACE REQUIREMENTS

3.1.1 Hardware Interfaces

User should have a basic computer with no extensive requirements. The project should be able to run on a standard virtual machine instance.

3.1.2 Software Interfaces

Web front-end using JavaScript or JavaScript variant.

3.1.3 User Interfaces

Angular2 - Single Page Application, or other similar variant

3.1.4 Other Communication Interfaces

Git, email, comments

3.2 SYSTEM FEATURES

3.2.1 General Website Features

- These may come free if we duplicate and edit the Federated Wiki.
- Material/otherwise nice looking design template
- User account management
- Create new account
- Edit existing account
- Login / logout functionality

3.2.2 Compile a Book

- Search functionality for finding existing textbooks
- Ability to duplicate and edit existing textbooks
- Add a new blank textbook

3.2.3 Textbook Compilation Interface

- Edit/New Chapter:
- Table of Contents view, with access to New Chapter and Edit Chapter
- Button to Add New Section/Scrap
- Button to Search Sections/Scraps
- User-friendly interface to add Section/Scrap to desired location in Chapter

3.2.4 Search

- By keyword, author, tag, field; ranked by relevance and/or community voting
- Scrap/Section search
- Textbook search

3.2.5 Publishing Interface

- Save textbook as PDF/send to printer
- Optional future feature: payment processor integration

3.2.6 Scrap Editor

- "Pretty" view for adding formatted text, media, etc to a scrap
- "Raw" view for managing scrap ordering
- Tag management (add, delete, search? tags)

4 APPENDIX A. GANTT CHART

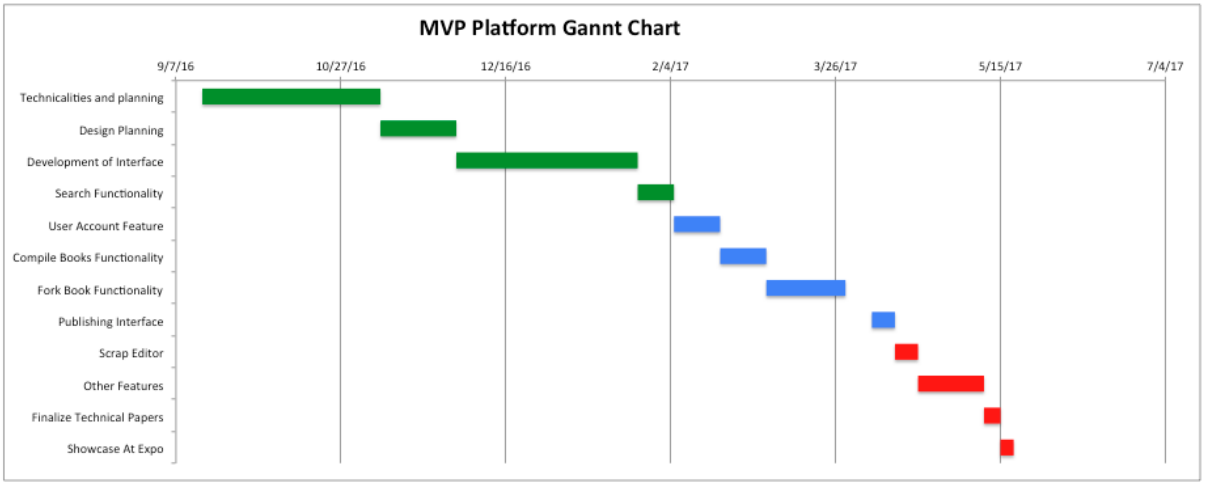


Figure 1. A simple preliminary Gantt Chart that outlines a rough sketch of our anticipated working time line

5 APPENDIX B. UNIFIED MODELING LANGUAGE DIAGRAM

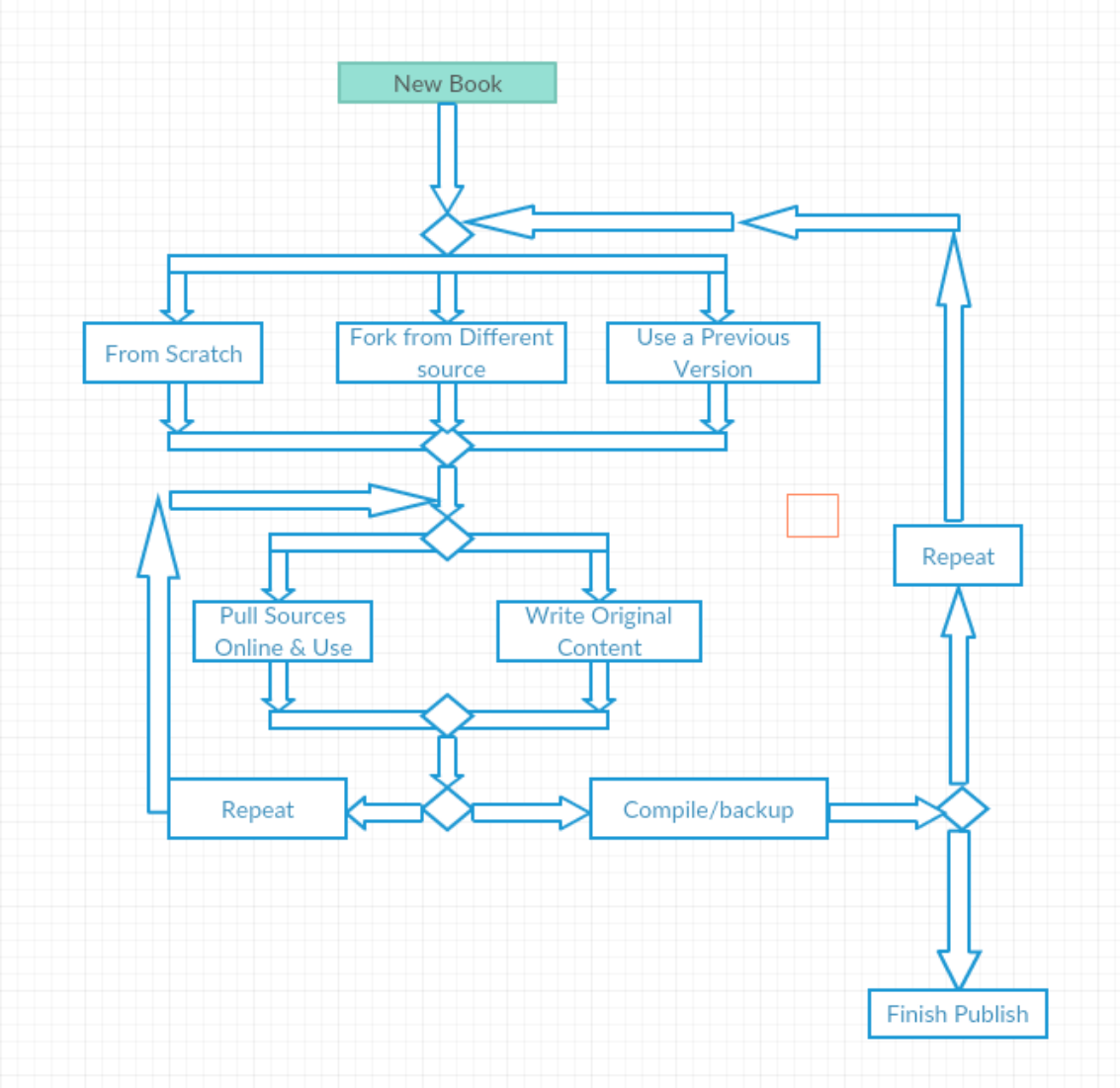


Figure 2. A preliminary UML Diagram that outlines the basic flow of information

REFERENCES

[1] W. Cunningham, “Smallest federated wiki,” http://wardcunningham.github.io/.

[2] Google, “Material design,” https://material.google.com//.

SIGNATURE PAGE

Dr. Carlos Jensen, Client _____ Date _____

Steven Powers, Developer _____ Date _____

Josh Matteson, Developer _____ Date _____

Evan Tschuy, Developer _____ Date _____

1 WHAT REQUIREMENTS CHANGED?

Throughout development of the Many Voices Publishing Platform, only a few changes were made throughout development that strayed from the original requirements and planning.

	Requirement	Description of What Happened	Comments
1	Operated via website	The product is to be operated via a website for wide availability and use	A website was used to demonstrate and allow for users to use the product.
2	Provide tools needed for textbook creation	The product will give the tools necessary for individual professors to create the textbook they need to match their desires	The product provides a variety of tools that facilitate the creation of a textbook with new or already existing materials that the user can combine in any way they see fit.
3	Document creation service	The product will be a document construction service	The product allows for the creation of documents.
4	Use of pre-existing materials	The product will allow for the creation of a textbook out of pre-existing materials (chapters / scraps), by searching and selecting from a list	The product allows for the use of pre-existing materials by searching and selecting. Also included is the ability to drag and drop selection.
5	LaTeX Support	The product will support the use of LaTeX within scraps	The product allows the user to enable LaTeX support when creating / editing a scrap.
6	Media Support	The product will support the use of media, such as images or perhaps video content	The product supports the use of images when creating a scrap.
7	Modern Design Paradigms	The product will be graphically driven by using a design philosophy such as Google's Material Design	The product employs the use of Materialize for the graphical user interface.
8	Security Barrier	The inter-process communication should be handled in a generic way that provides a high-security barrier, preventing malicious users from accessing other, non-specified applications	The product uses Google OAuth for user login and transmission of data and the product runs within a container, preventing external manipulation.
9	Will be a fork of Ward Cunningham's Federated Wiki	The product will use Ward Cunningham's Federated Wiki as the starting point for development	The product does not use Ward Cunningham's Federated Wiki as the base starting point. During development, a lot or problems surfaced from the lack of polish and features present within the Federated Wiki. Development started from the ground up instead.

10	Database	The product will use a database such as CouchDB 2.0, PostgresSQL, etc	The product features a backend Mongo Database.
11	Passwords	Passwords shall be stored in a securely hashed, non-reversible method, such that infiltration of the product by an unauthorized third party will not be able to result in the theft of user passwords	The product does not store any user passwords in any regards, and relies on Google OAuth for user authentication instead.
12	Performance & Compatibility	The product shall aim for 100 simultaneous users, as well as 95% uptime and usability. Additionally maintain 99.99% accessibility from major browsers	The final product does not promise specific response times, browser compatibility percentages, but maintains compatibility with the major web browsers.
13	Training Materials	The product will not feature any additional training materials beyond limited help screens built into the software and accompanying user manual	The product features tool tips instead of help screens.
14	Web Frontend using JavaScript	The product will use a JavaScript or JavaScript variant for the web frontend	The product uses the Aurelia JavaScript framework for the frontend.
15	User Interfaces	The product will use Angular2 for a Single Page Application or similar variant	The product uses Aurelia for the single page application instead of Angular2.
16	Compile A Book	Ability to compile a book, with search functionality for finding existing textbooks, ability to duplicate and edit existing textbooks, and add a new blank textbook	The product allows the user to search for existing materials, including textbooks, the ability to remix (fork and duplicate) a book, and create a new textbook.
17	Textbook Compilation Interface	Ability to edit / create a new chapter, a 'table of contents' view with access to editing. Button to add new scrap to a chapter, ability to search scraps, and user-friendly interface to add scrap to the desired location.	The product features a way edit and create new chapters, view a books 'table of contents' that allows the user to quickly edit or create a new chapter, and within editing a chapter, being able to edit or create a new scrap. The user also has the ability to arrange the chapters / scraps in the order they please. The product also features a user-friendly interface.

18	Search	The product will feature a way to search for relevant materials based on keyword, author, tag, field, ranked by relevance and/or community voting. Scrap, Chapter, Textbook searching	The product features a way to search for relevant materials based on keywords within a material, specific words applied with tags, that are automatically ranked by relevance. Being able to search by author is limited, because the end user is able to change their username.
19	Publishing Interface	The product will support the ability to save the textbook as a PDF / send the textbook to the printer.	The product features a way to preview the textbook in browser at the click of a button, or the user can view the textbook externally with the viewer of their choice. This allows the user to print the textbook if they desire.
20	Scrap Editor	The product will feature a 'pretty' view for adding formatted text, media, etc, to a scrap, and a 'raw' view for managing scrap ordering. Additionally tag management for a scrap.	The product features the ability to add LaTeX formatted text, or upload and add an image to a scrap. There is no 'pretty' and 'raw' view and instead both are seamlessly integrated.

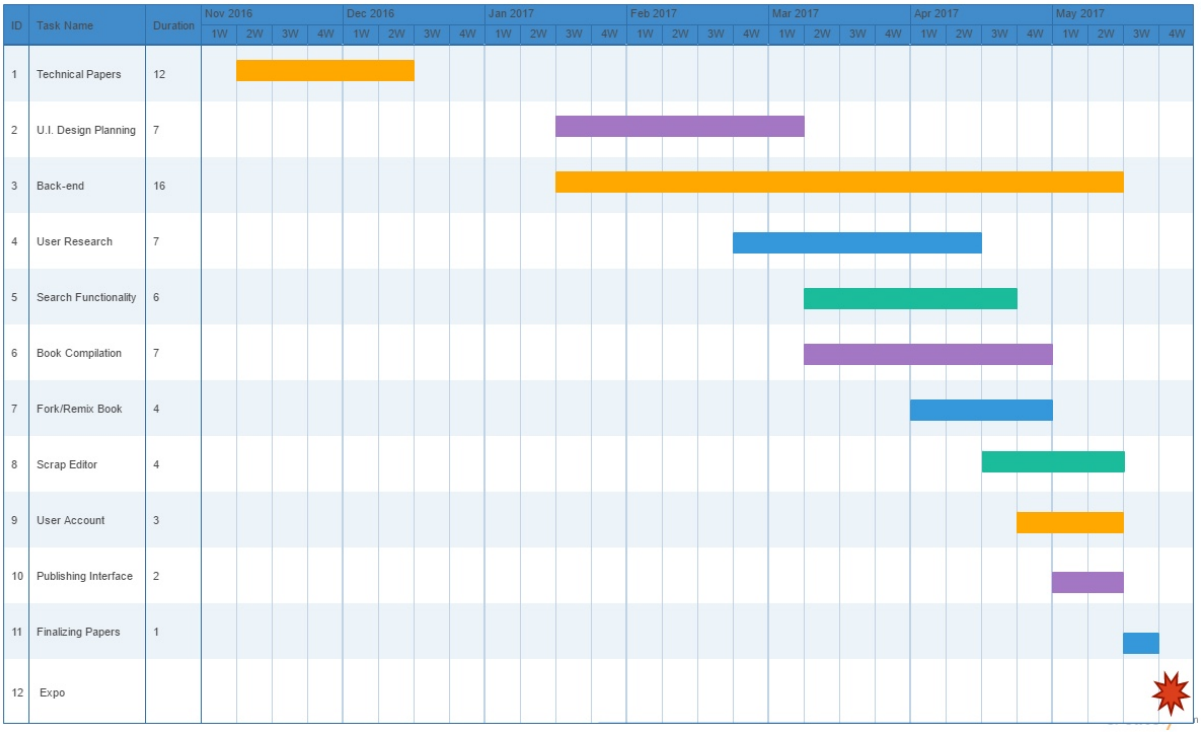


Fig. 1. The final Gantt Chart that outlines our working time line

Many Voices Publishing Platform

Software Design Document

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

Commix, Team 61

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. This document covers these technologies and follows the process that enable to the Many Voices Publishing Platform to succeed in delivering a working platform for textbook collaboration.

CONTENTS

1 OVERVIEW

The Software Design Document is a document to provide aid for the software development process by providing detailed information on how the software should be built. Additionally providing information on interactions between different pieces of software and users through use cases, UML diagrams, and other supporting information.

1.1 Scope

This Software Design Document is used to record design information and communicate design information to project stakeholders. This Software Design Document also provides the details of required functionality for the Many Voices Publishing Platform, a textbook creation platform for publishing textbooks.

1.2 Purpose

The purpose of this document is to describe the implementation of the Many Voices Publishing Platform (MVP Platform) software. The Many Voices Publishing platform is designed to allow for the creation of textbooks by College and University professors or any person interested in creating their own textbook.

1.3 Intended Audience

This document is intended for Professor D. Kevin McGrath, Dr. Kirsten M. Winters, and PhD Student Jonathan Dodge of Oregon State University for curriculum grading purposes. Additionally this document is intended for Dr. Carlos Jensen for the purpose of client information and senior capstone project purposes.

2 DEFINITIONS

Aurelia	A JavaScript client framework for mobile, desktop and web leveraging simple conventions and empowering creativity [1].

Alpha Test	Limited release(s) to selected, outside testers (Friends and Family)
Beta Test	Limited release(s) to cooperating customers wanting early access to developing systems (Professors and other educators)
Federated	Individual parts that stand as an individual but can be combined into a single unit
Final Test	Release of full functionality to customer for approval
PDF	Portable Document Format, that is able to combine text, graphics, and other information into a single document
PCI	Payment Card Industry, is a proprietary information security standard for credit cards in an effort to reduce credit card fraud
Scrap	A section of a textbook, which can contain formatted text (markdown or latex), and media
Section	An ordered collection of Scraps belonging to a chapter
Chapter	An ordered collection of Sections and Scraps
SDD	Software Design Document
SSRS	System and Software Requirements Specification
Source Control	An element of software design management, version control, and is the management of changes to documents, large web sites, computer programs, and other collections of data
Media	A standalone image, figure, or video. Can be embedded in a Scrap
Node	A JavaScript runtime designed to build scalable network applications
UML	Unified Modeling Language – A general purpose, development modeling language in the field of computer science
UI	User Interface – The means by which the user and a computer system interact, in particular the use of input devices and software
Web Application	An interactive program that can be accessed and is based through a web server instead of being stored on a user's desktop

Many Voices Publishing Platform

Wiki	A collaborative content editing platform

3 CONCEPTUAL MODEL FOR SOFTWARE DESIGN DESCRIPTIONS

3.1 Software design descriptions within the life cycle

The Software Design Description (SDD) is based in large part upon the System and Software Requirements Specification (SSRS) document. Requirements listed within the SSRS influence details within the SDD and the SDD may influence the SSRS details.

4 DESIGN DESCRIPTION

4.1 Introduction

When designing software to handle the creation of a textbook, the technologies in the background are equally as necessary as those in the foreground. The creation of a textbook requires various systems and technologies to handle the storing and presentation of data to allow the user to create their project.

4.2 Design Stakeholders

The stakeholders consist of Dr. Carlos Jensen, members of the Oregon State University senior capstone educational team, including Professor D. Kevin McGrath, Dr. Kirsten M. Winters, and PhD student Jonathan Dodge. Additional stakeholders include the development team consisting of Steven Powers, Evan Tschuy, and Josh Matteson.

4.3 Design Concerns

The design concerns for this project include the building of a User Interface with a functional JavaScript framework that allows for ease of use for users and developers.

User login and authentication will also be a design concern for this project, as preventing unintended access to another users work is very important.

Another concern consists of the usability of the interface and being able to inform the user of actions they expect to perform and can perform to complete their task of creating a textbook.

4.4 Design Views

The SDD will use UML diagrams to describe and visualize aspects of the design.

4.5 Design Viewpoints

This SDD will cover a number of different viewpoints, including: context, composition, logical, dependency, information, interface, and interaction viewpoints.

Context viewpoints cover the relationships, dependencies, and interactions between the system and its environment [2].

Composition viewpoints cover what information will be handled by the software.

Logical viewpoints cover what purpose the software will serve and how the software will achieve this purpose.

Dependency viewpoints cover outside elements that need to be integrated into the software in some way, as the implementation will depend on these outside elements.

Information viewpoints cover data that is present within the software or managed by the software in some way.

Interface viewpoints cover how designers and developers will be using the software, detailing the internal and external interfaces of the software.

Interaction viewpoints cover the interactions between different entities or elements within the software.

4.6 Design Elements

Design elements within our software will include a variety of different features that are often considered standard elements within software. These elements include buttons, text boxes, search boxes, menus, and clickable links just to name a few. The menus of the system will be limited for user convenience and will provide a meaningful icon or text representation for quick affordability for the user. Within the text editing area, the user will be able to arrange text how they would like it to appear in a finalized—compiled version.

The text area will also allow users to specify other documents to include, which will be handled by the software in the background at time of compilation. Each included document or file will be stored as a separate document with version control capabilities.

4.7 Design Rationale

For this project, design choices are made based on client specifications as well as development concerns due to technology availability and adaptability to the current system. Our client Dr. Carlos Jensen wants the project to allow for the easy creation of textbooks through what is called the Many Voices Publishing Platform. Design choices will be made to accommodate this requirement.

4.8 Design Timeline

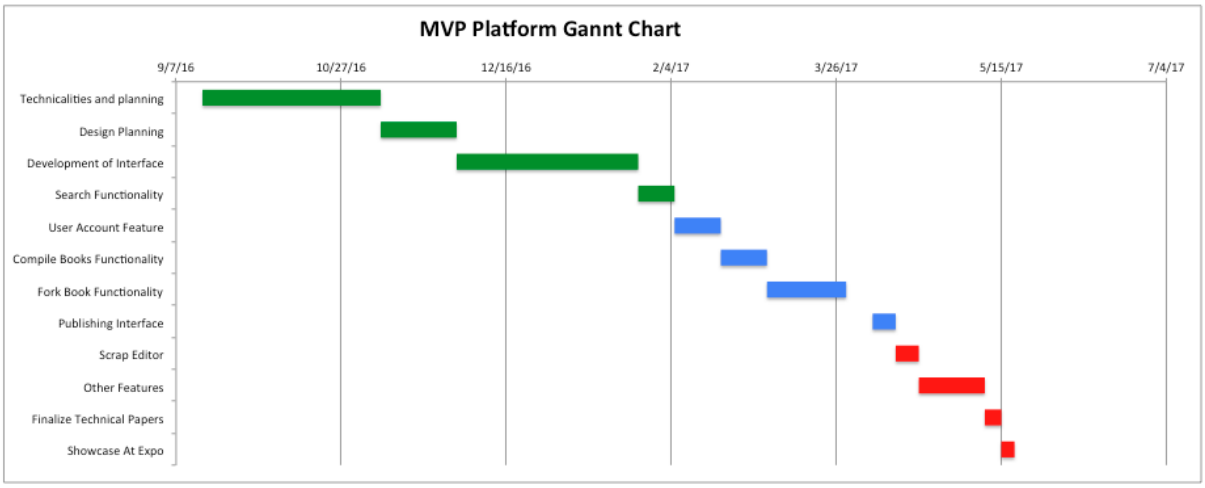


Fig. 1. A preliminary Gantt Chart that outlines a rough sketch of our anticipated working time line as of Fall term

4.9 Design Languages

This document will use UML as the design language.

5 DESIGN VIEWPOINTS

5.1 Introduction

This section will cover: context, composition, logical, dependency, information, interface, and interaction viewpoints.

Steven Powers is covering User Interface Tools, User Login & Authentication, and Interface Design.

Evan Tschuy is covering Server Back-end, Text Formatting, and Password Storage.

Josh Matteson is covering Testing, Revision Control Software, and Database.

5.2 Viewpoint: User Interface Tools

By: Steven Powers

5.2.1 Interface

The user interface is one of the most important parts to the Many Voices Publishing Platform. An easy to use UI can make the difference between two competing software solutions. The Many Voices Publishing Platform will be interacted through a website that will display a users documents and their current document. The User Interface Tools will allow for a high quality user experience with a high number of screen repaints per second, further increasing the ease of use with the software through a fluid user interface [3].

5.2.2 Design Concerns

A poorly implemented UI can result in users choosing a competing product or simply deciding not to use any software for their intended purpose. Users are often impatient and quick to abandon software, further proving the need for a robust and easy to use User Interface Toolset.

5.2.3 Design Elements

The User Interface Tools will allow for scalability when it comes to using the software on different platforms, including mobile, and desktop environments. Additionally the tools will provide great interact-ability for the user.

5.2.4 Function Attribute

This component provides the user interface for users to interact with while using the software. Handles display of information and provides the interface for input.

5.2.5 Relationship

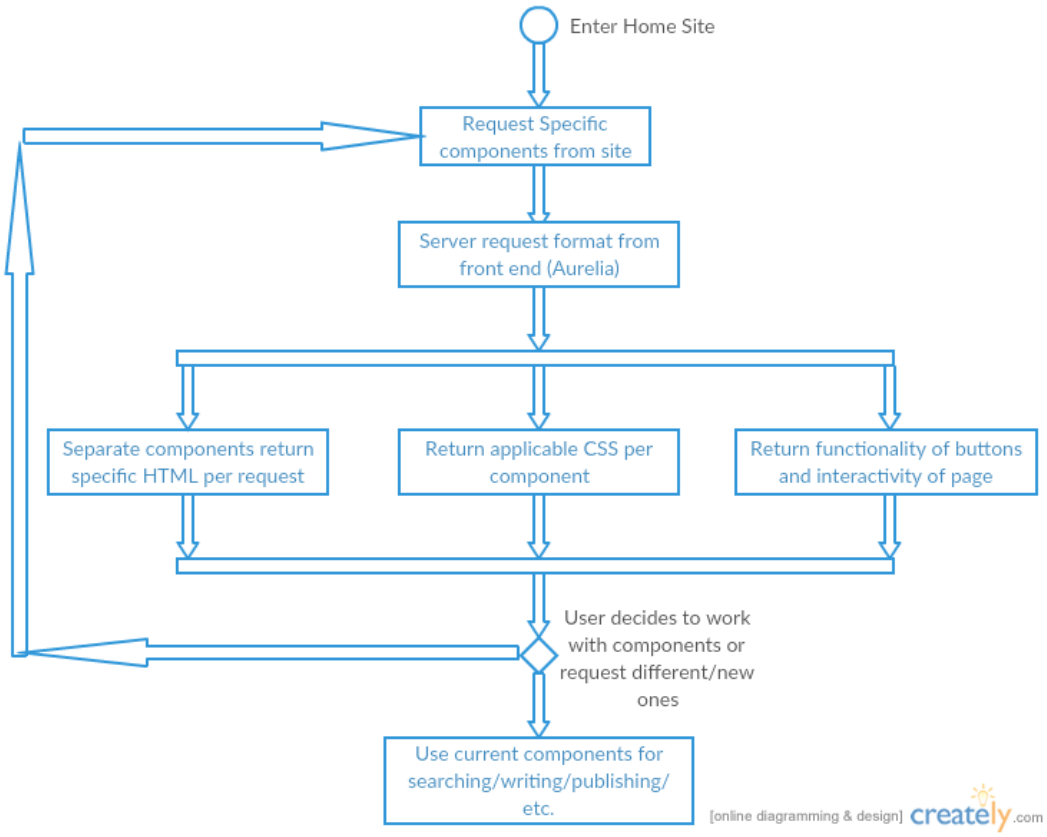


Fig. 2. A preliminary UML diagram of how the user interface tool of Aurelia will enable transmission of data to the screen or from the screen, up to date as of Fall term

5.3 Viewpoint: User Login & Authentication

By: Steven Powers

5.3.1 Context–Dependency

A user login system is standard affair for most websites on the Internet. How these user logins take place and authenticate users can vary quite significantly depending on the implementation. User login security is important for protecting the customer as well as the reputation of the software and company. User Login & Authentication are dependent on Evan Tschuy's section on Password Storage section.

5.3.2 Design Concerns

Handling user logins and user authentication can be quite a painstaking process, as any mistake can cost you customers and any reputation that was present before the mistake was exploited. Authentication, or the matching of user submitted data with our stored credentials can be exploited though a simple MySQL command, if our servers are not secured properly. In house solutions can be buggy, or not as secure. Third party solutions require an account with those services and has security in their hands.

5.3.3 Design Elements

A way to login securely, through created credentials or through a third party login system, such as Login with Facebook or Google.

5.3.4 Function Attribute

This component provides the functionality of user login process and user authentication within the software.

5.4 Viewpoint: Interface Design

By: Steven Powers

5.4.1 Information

The approach for user interface design is quite different than that of the tools being used. Interface Design refers to the methodologies employed to create the UI. This often takes the form of user studies, and demoing of prototypes and release candidate mockups for feedback. For our software, our target audience is professors, especially those interested in publishing their own book currently or in the near future. Using the target audience as a design requirement, the designer is able to gleam a lot of information about how to best serve this user. Methodologies include user centered design, activity centered design, and self design principles to list a few common disciplines.

5.4.2 Design Concerns

Interface Design is an often overlooked portion of any software product. For some software products it would be no surprise that the software is never used in house, we are trying to avoid this feeling. User centered design, while often the standard for the Computer Science industry, it very costly, both in terms of time and money. There is a large amount of time into user research studies and live demos. Self design, while much faster, and easier to perform, can lead to results that do not satisfy your users expectations.

5.4.3 Design Elements

An Interface Design methodology that allows for efficient use of time as well as successful design choices to best suit our users.

5.4.4 Function Attribute

Provides methodologies for improving Interface Design to assist users and developers.

5.4.5 Relationship

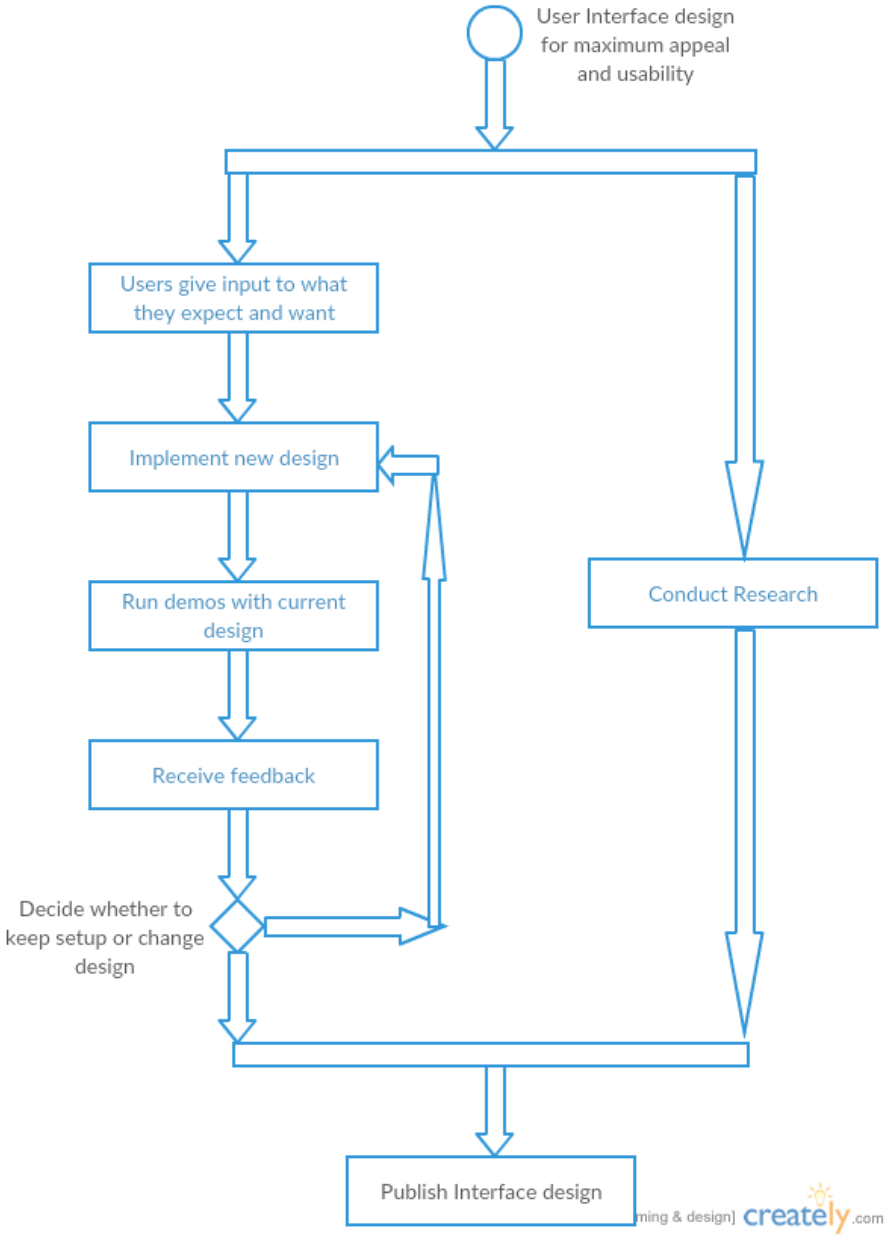


Fig. 3. A preliminary UML diagram of how user centered design & user research can improve a UI for increased affordance and usability for the target audience, up to date as of Fall term

5.5 Viewpoint: Server Back-end

By: Evan Tschuy

5.5.1 Backend

A standard web application follows more or less a standard design flow. When a request is received, a URL parser parses the URL and passes it to the appropriate function, along with all of its parameters. The function then operates on the data somehow, and returns either the result of a template render or a block of data in JSON/XML to be returned to the client. Inside the function, the heavy lifting of data manipulation, storage, etc. takes place.

5.5.2 Design Concerns

The main concern for the backend of the project is how the back end will communicate with the version control system. For instance, building on top of Git, it is necessary to also verify that the backend language chosen has a library that can be used to easily interact with Git. Then, it will be necessary to build an internal library that can be placed on top of Git that exposes only the operations needed for the textbook project.

5.5.3 Design Elements

The backend design, especially the layer interacting with Git, will play a critical role in speed of development. By implementing a Snippet and Textbook super-layer on top of the existing Git library, we can eliminate having to think about Git as early as possible, and spend our time instead on interacting with Snippets and the Textbooks.

5.5.4 Function Attribute

This function provides the base on which the rest of the project is built the interaction layer between the frontend and the revision control system.

5.5.5 Relationship

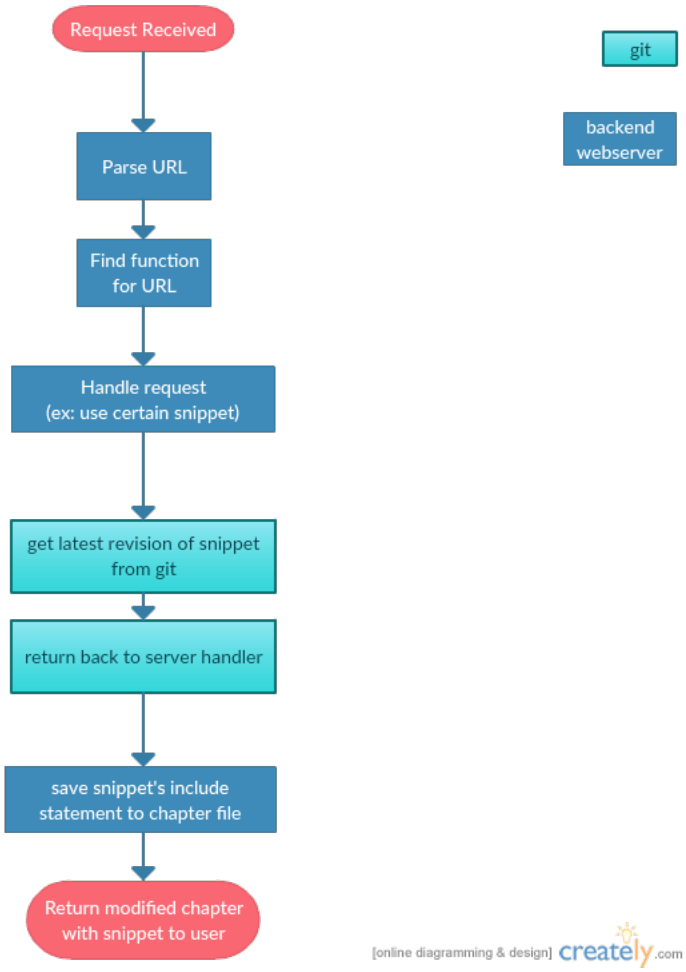


Fig. 4. An example backend handling flow. The application determines which function to call, and the function handles the request and sends any necessary file operations to the git module.

5.6 Viewpoint: Text Formatting

By: Evan Tschuy

5.6.1 Formatting

The front end of the platform lets users interact with "snippets" of documents a block of text that, paired with potentially dozens of other snippets, can make a chapter. Then, chapters need to be combined together to make a full textbook. Using a LaTeX backend, snippets can be related to an

`\input{}`

command, which takes the raw commands of a document and puts them into another document. Chapters can be related to

`\include{}`

commands, which start a new page and add the new section. In this way, we can have one file for the table of contents, which controls which chapters are included in which order, and one file for each chapter, which controls which snippets are included in which order.

5.6.2 Design Concerns

The files referenced are all stored in version control, as discussed below. Therefore we need to relate the files to a specific version as stored. Additionally, it needs to be entirely invisible to users how the text is being split the users should not know whether the backend is written in LaTeX using include and input statements, or in Markdown with string concatenation, or any other possible implementation.

5.6.3 Design Elements

An abstraction of snippets and chapters into LaTeX documents in such a way as to make manipulation of them individually and as a whole simple.

5.6.4 Function Attribute

This component provides the functionality of document generation, storage, and manipulation.

5.6.5 Relationship

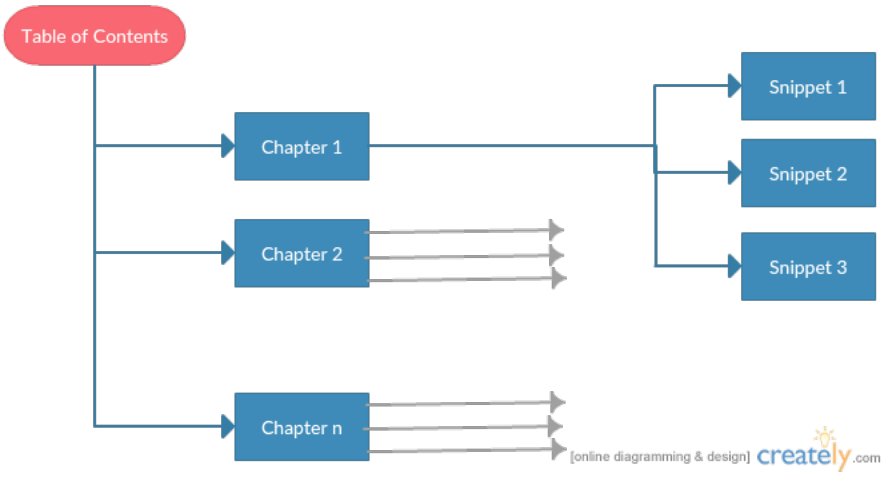


Fig. 5. A document is divided into chapters (which are included under a table of contents) and snippets (which are parts of a chapter).

5.7 Viewpoint: Password Storage

By: Evan Tschuy

5.7.1 Password hashing

Any time a password is being requested from a user, it should be securely and irreversibly hashed. By securely hashing a password, it becomes impossible for hackers with access to the user database to use stored credentials to compromise user accounts on other sites even if users reuse passwords between sites. A well designed cryptographic hash, such as bcrypt, includes a salt value, which is simply added to the beginning of the password at hash time, stored in the database in plain text, so that two uses of the same password result in different hashes.

5.7.2 Design Concerns

A secure password storage system always hashes passwords, and requires hackers to crack each password individually by guessing passwords one by one. Another alternative to storing user passwords at all is, as discussed above in the user authentication section, to use a third-party user authentication service such as those offered by Google, Facebook, etc. For a platform used by professors, who may

not all have a Google or Facebook account, however, it should always be possible to create an account that does not require a third-party account.

5.7.3 Design Elements

A well-designed, secure system in which any password stored cannot be reversed without a slow cracking process. Preferably, users would not store any passwords at all with the service and instead would use third-party authentication.

5.7.4 Function Attribute

This component provides the functionality of secure password storage.

5.7.5 Relationship

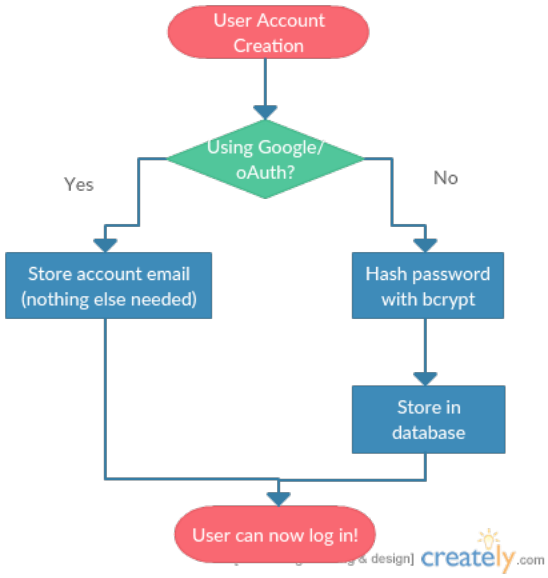


Fig. 6. Passwords will either be securely hashed, if the user is not using external authentication, or not needed at all, if the user is using external authentication.

5.8 Viewpoint: Testing

By: Josh Matteson

5.8.1 Information

Testing is one of the most substantial parts when considering the success and thoroughness of an application. If not properly tested, the application will contain numerous bugs resulting in: faulty functionality, unpredictable behavior, and the pages not loading.

5.8.2 Design Concerns

The main concerns with working with a testing framework is the level of complexity and with that the learning curve. Jasmine, existing for almost a decade, has had numerous contributors to its documenation. This ensures that, while making progress, we won't be short of useful information in the furthest reaches of the internet.

5.8.3 Design Elements

Jasmine is open source, and therefore, will be used as the main testing framework for application. It will be one of the developer dependencies and will mostly interact with the existing javascript/typescript of our application. Jasmine will be ran through our node express backend, and will come with a coverage report for our app. Whether this is from Jasmine or from another testing framework that works closely with it.

5.8.4 Function Attribute

Assist in development and thoroughness of the application. Will drastically reduce bugs.

5.8.5 Relationship

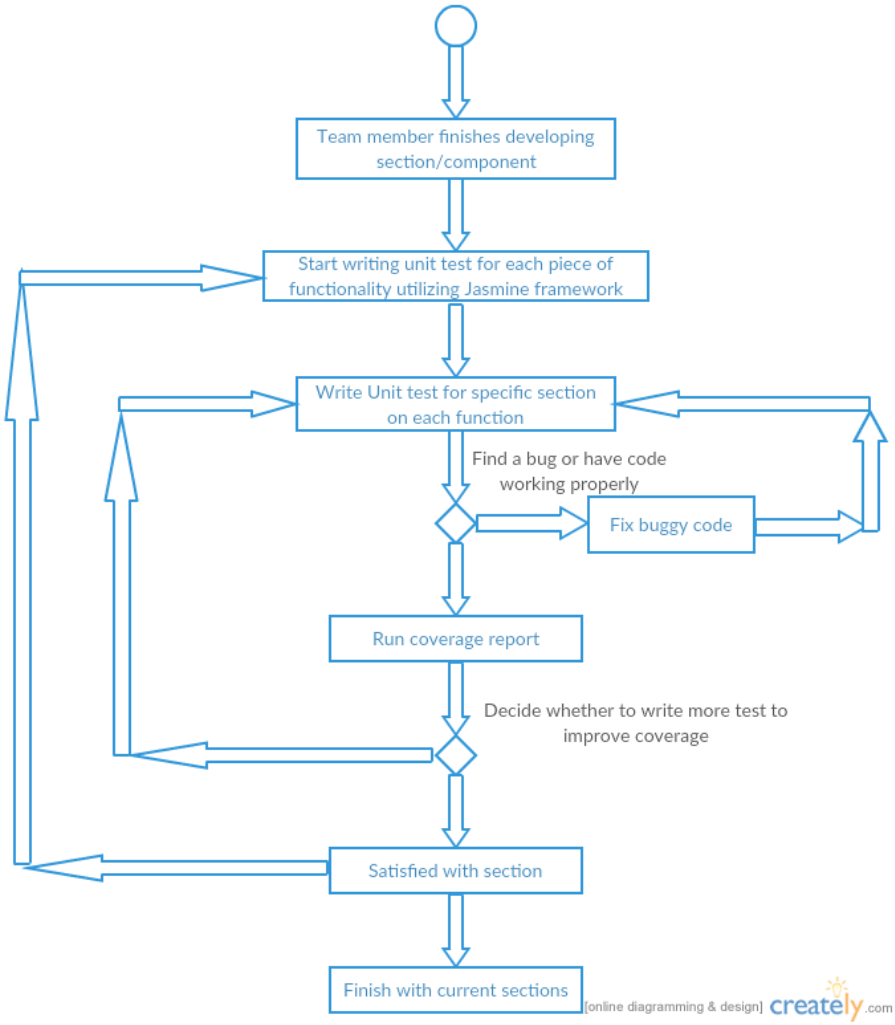


Fig. 7. A preliminary UML diagram demonstrating the flow of testing with Jasmine

5.9 Viewpoint: Revision Control Software

By: Josh Matteson

5.9.1 Revision Control

One of the more important elements of our application, the revision control will let the user have full control over their current work and back ups. If the user realizes that a section they wrote conflicts with something someone else wrote, they can then backup from a previous version. This will all be done with Git as the background technology, which in essence, is the backbone of major web platforms like Github. This component will give users endless control with branches, previous versions, and much more.

5.9.2 Design Concerns

One of the main design concerns is properly integrating Git with the rest of the application. Ensuring that the flow from user to Git then to database will require well force all parts of the app to be working.

5.9.3 Design Elements

Git will act as middleware in between the front end and the backend, this will be taken into mind when moving forward. The user will use one of the front end functions, which will request an action of Git, and finally perform database operations.

5.9.4 Function Attribute

The overarching purpose of Git is to give users free reign to write whatever they want with little to no consequences from themselves or other users.

5.9.5 Relationship

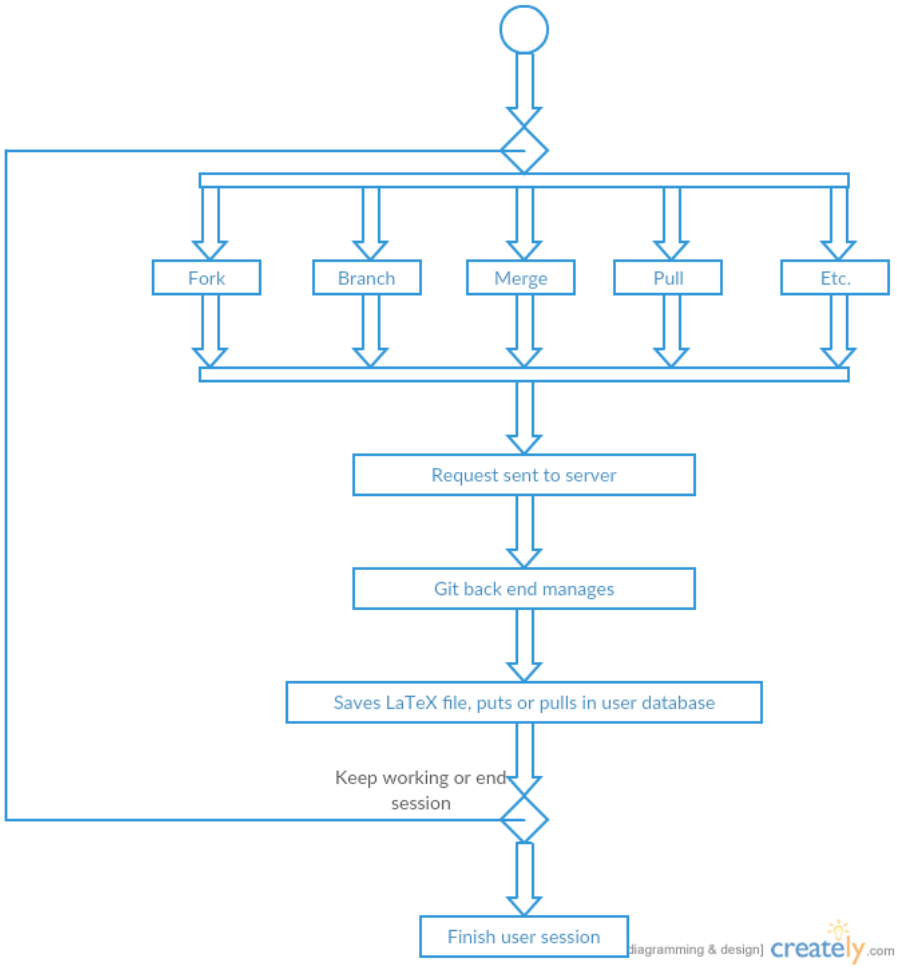


Fig. 8. A preliminary UML diagram demonstrating the relationship of Git to the request of the user

5.10 Viewpoint: Database

By: Josh Matteson

5.10.1 Database

While understanding what a database does is more common even among non computer science majors, the aspects of its relation to other components of our application are not self explanatory. The database's most important role consist of storing user information, and ensuring that information does get misplaced or skewed a long the way to the user.

5.10.2 Design Concerns

As with most databases, one of the main design concerns is the complexity of the calls to it. Some databases use a query structure, other ones use explicit calls. SQL Server, as is part of the name, will be using query calls to receive information.

5.10.3 Design Elements

Our backend components, being Node.js and Express, will be doing the calls to the database. Front end functions will call general purpose functions in the backend, which will then make a call to the database. This will then go back up the ladder to the user.

5.10.4 Function Attribute

The database stores all important information about the user and their revisions, passwords, and preferences.

6 ANNEX A - (INFORMATION BIBLIOGRAPHY

REFERENCES

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

[2] R. Woods, "Viewpoints," http://www.viewpoints-and-perspectives.info/home/viewpoints/.

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

[4] W. Cunningham, "Smallest federated wiki," http://wardcunningham.github.io/.

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

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

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

[8] Facebook, "Facebook login for apps," https://developers.facebook.com/docs/facebook-login/overview.

[9] OpenID, "Openid the internet identity layer," http://openid.net/connect/faq/.

[10] C. Bowles, "Looking beyond user-centered design," http://alistapart.com/column/looking-beyond-user-centered-design.

[11] U. D. of Health & Human Services, "User-centered design basics," https://www.usability.gov/what-and-why/user-centered-design.html.

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

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

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

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

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

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

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

[19] CoderWall, "Javascript test framework comparison," https://coderwall.com/p/ntbixw/javascript-test-framework-comparison.

[20] Microsoft, "Platform for intelligent applications," https://www.microsoft.com/en-us/sql-server/.

[21] MongoDB, "Mongodb — for giant ideas," https://www.mongodb.com/.

[22] MySQL, "Mysql," http://www.mysql.com/.

[23] I. B. Network, "Why are we excited to talk about mongodb?" http://www.ibmbpnetwork.com/linux-blog/excited-about-mongodb.

7 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.

8 SIGNATURE PAGE

Dr. Carlos Jensen, Client _____ Date _____

Steven Powers, Developer _____ Date _____

Josh Matteson, Developer _____ Date _____

Evan Tschuy, Developer _____ Date _____

1 WHAT DESIGN DECISIONS CHANGED?

Throughout development of the Many Voices Publishing Platform, only a few changes were made throughout development that strayed from the original requirements and planning.

	Design	Description of What Happened	Comments
1	User login & Authentication	We were unsure if we would be using 3rd party login or rolling our own; we used Google's oAuth backend	3rd party login allowed us to simplify the new account creation experience.
2	Book compilation	LaTeX files are constructed in-memory and saved to disk before compilation, rather than being saved individually and constructed with `\include` and `\input`.	This allows us to more easily change the document design and to implement an easier data flow.
3	Make text formatting transparent to user	Users can choose to write scraps with full LaTeX enabled, or in plain-text mode.	This change allows for much more powerful document creation. Creating a table or a chart is now possible, which would not have been possible if the user had to use a GUI-based editor.
4	Use passwords to authenticate users	Instead of passwords, a user logs in with their Google account. A token is generated that is then stored in their browser and in the backend database.	Users no longer need to remember a password and instead of storing a password, we can store unique, randomly generated tokens, removing the risk of password leakage.
5	Passwords must be salted and hashed	Instead of passwords, tokens are used. Tokens are hashed but not salted to allow for token lookup directly.	Tokens do not need to be salted, as they are already unique to the service and leaking a token does not harm user security. Tokens are stored salted, and so still require a large amount of computation to break a single token.
6	Use a SQL database for user data	Instead of a SQL database, a document storage database was used.	Using a document storage database made sense, since the data we were storing was already created as document-style JSON objects.

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 Powers

1.2.1 User Interface Tools

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.

1.2.1.1 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.

1.2.1.2 Evaluation Criteria:

The options are evaluated on

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

1.2.1.3 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

1.2.1.4 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.

1.2.1.5 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 User Login & Authentication

Option 1 - OpenID Connect

OpenID Connect allows for clients of all types, including browser-based JavaScript and native mobile apps, to launch sign-in flows and receive verifiable assertions about the identity of signed-in users [6].

Option 2 - Facebook

Facebook Login for Apps is a fast and convenient way for people to create accounts and log into your app across multiple platforms [7].

Option 3 - PHP & SQL

Using PHP and SQL to compare submitted usernames and passwords against stored data on a database.

1.2.2.1 Goals:

An efficient and secure method for allowing for users to login and continue editing their documents from any computer or device they choose.

1.2.2.2 Evaluation Criteria:

The options are evaluated on

- Ease of Use
- Features
- Security

1.2.2.3 Option Comparison:

	OpenID	Facebook Login	PHP & SQL
Ease of Use	Requires Credentials with Corresponding Login Providers, Lots of available libraries	Requires Credentials with Facebook and an App ID with Facebook	Easy to implement, but if setup incorrectly can lead to problems
Security	Relies on credential host and user security	Relies on Facebook and user security	Relies on password protection implementation and user security
Features	Easily log in with OpenID partner credential hosts (Google, Microsoft, Yahoo, etc)	Easily log in with Facebook credentials	Easily log in with user created account and password

1.2.2.4 Discussion:

The ideal user authentication system would be a combination of all three of the above implementations. While logging in with Facebook would make it easier to determine who is using the service, preventing unauthorized users from accessing unreleased copyrighted material, not everyone has a Facebook. Additionally using an OpenID login system would be reliant on other platform holders that use OAuth 2.0. Finally, using a self created account is often the easiest and can allow users to not be tied to a given account and also prevent private information from being retrieved from user accounts.

1.2.2.5 Selection:

For our implementation, we plan on using initially a PHP and SQL system to validate user account information on our database. Additionally, we will look into adding both OpenID and Facebook Login down the road.

1.2.3 Interface Design

Option 1 - User Centered Design

A deep understanding of the target audience is able to provide insights into how to design and develop your application to suit your intended users [8].

Option 2 - Activity-Centered Design

Instead of focusing on research about intended users, the design and development are focused around making a given activity logically designed [9].

Option 3 - Self Design

The designer is responsible for representing the target audience. Though this can be a poor representation of the intended audience [9].

1.2.3.1 Goals: A design principle that allows for user interfaces that lead to user interfaces that are accepted by users and are easy to understand.

1.2.3.2 Evaluation Criteria:

The options are evaluated on

- Ease of Use
- Strengths
- Weaknesses

1.2.3.3 Option Comparison:

	User Centered Design	Activity-Centered Design	Self Design
Ease of Use	Long process, that takes a lot of data gathering to provide insights into a target audience.	Easier to design an activity when not trying to cater a specific audience.	Very easy to design what works well for you as a designer.
Strengths	Allows for the designer to understand what makes a user think the way they do. This allows for an interface design to be molded to an expected user.	Allows the designer to design a user interface based around an activity that a user will be performing instead of designing to a users wants and desires.	Allows for easy creation of user interface of how the designers see fit. Perfect for a target audience that is just like the designer.
Weaknesses	Takes a long time to gather enough information to be able to design a good solution that feels natural to a target audience user.	Designed interface might work well for an intended activity, but could be antagonistic to a target audience.	Interface could be intended for an entirely different audience, leaving a confusing experience.

1.2.3.4 Discussion:

The MVP Platform is highly user focused, which initially led the team to decided on User Centered Design. Activity-Centered Design or Self Design would greatly reduce the burden of research and discovery into what our target audience would like to see or be comfortable with naturally. Activity-Centered Design, if performed properly would result in interfaces that clearly work as intended, though might be off putting to our users. Self Design would allow for one of the team members to decide how a certain element shall look, but again can fall into an interface that does not satisfy our users.

1.2.3.5 Selection:

For our implementation, we plan on using User Centered Design. This is because users are our very important for our project. If our users do not like our user interface, then they will be less likely to use our software.

1.3 Josh Matteson

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]

1.3.1.1 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.

1.3.1.2 Evaluation Criteria:

The options are evaluated on

- Ease of Use
- Features
- Documentation
- Integration
- Other

1.3.1.3 Option Comparison:

	Mocha	QUnit	Jasmine
Ease of Use	Language is like spoken language, which makes for easier to understand	Not as friendly, language is similiar to other basic Unit testing frameworks	Language is like spoken English, fairly easy to understand
Features	Spys, mocks, stubs, callbacks, etc. Most features	Most testing is done through assertions only, not as many features	Spys, mocks, stubs, callbacks, etc. Lacks Assertions, but can be implemented with library
Documentation	Fairly common, moderate amount of documentation	Not well documented	Most popular, abundant documentation
Integration	Moderate difficulty to set up	Easiest to set up	Moderate difficulty to set up
Other			Familiar

1.3.1.4 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 these mentioned testing frameworks: Mocha, QUnit, and Jasmine.

1.3.1.5 Selection:

Jasmine, even though more difficult to integrate and set up, would be most feasible. Jasmine has many features that win out over Mocha and QUnit, as well as thorough documenation. The test are easily grouped through describe blocks, which makes it easy to identify where certain problems are. There is familiarity with Jasmine over the other testing frameworks, which means less time learning for the team.

1.3.2 Revision Control Software

Option 1 - Git

Git is an open source, free, distributed version control system. It's designed to handle small to very large projects with speed and accuracy. It is commonly used by a variety of companies. [13]

Option 2 - CVS - Concurrent Versions System

CVS is the most popular and widely adopted revision control system to date. It is widely considered because of its low learning curve. [14]

Option 3 - SVN

Subversion is another open source version control software used by platforms like Ruby, Python Apache, and more. SVN has many versions and IDEs available. [15]

1.3.2.1 Goals: Our application will widely revolve around the revision/version control software that we use, which makess choicing the correct one ideal. The main goals for our revision control would be ease of incorporation into our app, ease of use, and the features it comes with. More criteria found on the next page.

1.3.2.2 Evaluation Criteria:

The options are evaluated on

- Established
- Features
- Ease of Use
- Speed

1.3.2.3 Option Comparison:

	Git	CVS	SVN
Established	More than a decade	More than three decades	Around 15 years
Features	Access to tree offline, highly distributed peer-to-peer model	Basic Features	Atomic Operations, Various IDE Plugins
Ease of Use	Steep learning curve	Relatively low learning curve	Relatively low learning curve
Speed	Very fast	Very slow	Slow

[14]

1.3.2.4 Discussion:

Revision control will be used frequently in our app to save progress and ensure progress hasn't been lost. This in mind, we need software that ensures speed and can perform various operations inexpensively. While most of the revision control software offers features, only one of them offers the speed required for frequent revisions.

1.3.2.5 Selection:

Although accompanied by a steep learning curve, Git offers all the features and speed required for the operations of our application. Fortunately, most of our team members are familiar with using Git to one extent or another. This makes the trade off feasible.

1.3.3 Database

Option 1 - SQL Server

SQL Server is owned by Microsoft, and is known for its scalable, hybrid database platform. Being owned by Microsoft, this database is well known. [16]

Option 2 - MongoDB

MongoDB is a document based database, with an Expressive Query Language and Secondary Indexes. [17]

Option 3 - MySQL

Many of the worlds largest companies rely on MySQL, such as Facebook, Google, and others. It is a relational database. [18]

1.3.3.1 Goals: Having an understandable, well built database can aid in the flow of building an application as well as delivery. This in mind, we want to balance cost with effectiveness.

1.3.3.2 Evaluation Criteria:

The options are evaluated on

- Cost
- Type
- Ease of Use

1.3.3.3 Option Comparison:

	SQL Server	MongoDB	MySQL
Cost	Enterprise: $12,256, Standard: $3717, Developer: Free	OpenSource, but essential periphery software can range from $45 a month to as much as $5225 a month	Enterprise: $5000, Standard: $2000, Community Edition: Free
Type	Structured Query Language	Expressive Query Language	Structured Query Language
Ease of Use	Uses Queries to gather information	EQL data access and manipulation in sophisticated ways, operational and analytical applications [19]	Uses Queries to gather information

1.3.3.4 Discussion:

The most obvious and transparent qualities for a database are the cost and type, however, databases that come with a free edition allow for us to test and experiment while developing. This is imperative to the development process, and shouldn't be overlooked. Allowing for an easy transition from the free edition to a standard or express edition is important as well. A non relational database like MongoDB could be beneficial when not knowing the type of data that will be stored.'

1.3.3.5 Selection:

All things considered, SQL Server would be considered our top pick. Experience using an SQL based database before aids in the difficulty of using a query. Having a free edition contributes a moderate amount into the decision as well.

1.4 Evan Tschuy

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.

1.4.1.1 Evaluation Criteria:

The options are evaluated on

- Ease of use
- Language
- Features
- Ecosystem
- License

1.4.1.2 Option Comparison:

	NodeJS	Django	Flask	Ruby on Rails
Ease of Use	Javascript backend shares language with frontend; super quick iteration	Large framework containing all needs within, with a large learning curve	Microframework containing only minimal needs; requires finding external packages	Old framework with massive, but aging, ecosystem
Language	Javascript	Python	Python	Ruby
Features	Minimal HTTP interaction, massive external ecosystem for extras	Massive built-ins	Minimal with external ecosystem for extras	Massive built-ins
Ecosystem	Massively popular today with expansive and growing ecosystem	Decently large ecosystem with built-ins for most tasks	Decently large ecosystem	Large ecosystem but fairly old
Release	7.1.0	1.10.3	0.11	5.0.0.1
License	MIT	BSD	BSD	MIT

1.4.1.3 Discussion:

All backends listed are popular within their respective communities. However, more new projects are being created using NodeJS, as its modern paradigms and sharing of a language with frontend development allow developers familiar with it to iterate quicker and write more expressive code. Django's built-ins allow for a quicker initial development time but mean being isolated from the rapidly expanding ecosystem around NodeJS. Ruby on Rails is a rather old backend, and has not shown to have the modern flexibility of Node.

1.4.1.4 Selection:

As NodeJS has the most expansive ecosystem, and allows us to share a common language between the front and backennds of the project, we will be using it over the other options considered. Additionally, Ward Cunningham's Federated Wiki uses it as one of its backends, and if we fork it, we can continue to use its NodeJS backend.

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 or videos; 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.

1.4.2.1 Evaluation Criteria:

The options are evaluated on

- Ease of use for end-users
- Markup options
- Compile language
- Security

1.4.2.2 Option Comparison:

	Markdown	ReStructured Text	HTML
Ease of Use	Easy to use, with minimal options and human-readable markup; different implementations have slight differences leading to confusion	Relatively human-readable markup but with massive number of options	Essentially infinite options but not very human-readable/human-writeable
Markup	Options readable in single-page document, not allowing for high flexibility	Highly featureful with well-defined language	Allows for infinite options along; can use CSS to fine-tune display
Compile Language	Dozens of different libraries for different languages, each with slightly different interpretation of the markup	Python	No compilation needed to display but some backend processing needed for security
Security	Small language leads to minimal exploits	Well-defined with real-world-tested libraries	Needs careful processing to stop end-users from inputting harmful raw input (including scripts)

1.4.2.3 Discussion:

All languages listed allow users to do simple things like bold text and link to other pages. However, HTML offers the most flexibility and allows users to be able to do anything they want. Allowing this while maintaining ease-of-use would require a frontend library that can allow users to interact with the document in ”what you see is what you get”, or WYSIWYG, mode.

1.4.2.4 Selection:

The Federated Wiki uses HTML with minimal security processing. If we fork the Federated Wiki, we will use HTML with some added processing to increase its security. Otherwise, for ease of implementation, we will use Markdown for its backend language support, but implemented in such a way as to allow easy replacement of the code with some other markup language as wanted.

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

Another option for password storage is to store the passwords in plain text. This allows users to recover their passwords directly through a password reminder email. However, this comes with the major downside that compromising the database allows a hacker to be able to access any accounts on unrelated services where users use the same username and password (a common pattern in non-technical and technical users alike).

1.4.3.1 Evaluation Criteria:

The options are evaluated on

- Cracking
- Storage
- Resistance to Hacking

1.4.3.2 Option Comparison:

	Bcrypt	Scrypt	pbkdf2	plain text
Cracking	Bcrypt is highly resistant to cracking on CPUs and GPUs, but can be cracked quickly using specialized FPGAs.	Scrypt is highly resistant to cracking on CPUs, GPUs, etc but takes a large amount of time to verify a valid password.	pbkdf2 is highly resistant to cracking on CPUs but can be easily cracked on a GPU.	A plaintext password does not need to be cracked as it is already stored as a raw password.
Storage	can be stored in a database without issue.	can be stored in a database without issue.	can be stored in a database without issue.	Plain passwords must be stored in a way that ensures they can never be hacked, which is impossible.
Resistance to Hacking	A bcrypt password hash must be cracked before it can be used elsewhere.	Scrypt hashes must be cracked before use elsewhere.	pbkdf2 hashes must be cracked before use elsewhere.	A plain password, once retrieved from a database, can be used along with the username/email associated to hack other sites.

1.4.3.3 Discussion:

Password storage is a tradeoff between ease of use and difficulty of reversing. Scrypt is too slow for use on a website with many users, whereas plain text passwords are too insecure as a hacker can reuse the password immediately without cracking. pbkdf2 and bcrypt do a good job defending against CPU cracking, but as pbkdf2 can be cracked using a GPU, bcrypt is left remaining as the best tradeoff between speed and cracking.

1.4.3.4 Selection:

As mentioned above, bcrypt-hashed passwords present a good tradeoff between cracking ability and verification speed. As such, the Many Voices Platform will use bcrypt to securely verify any password used with the system.

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] OpenID, "Openid the internet identity layer," http://openid.net/connect/faq/.

[7] Facebook, "Facebook login for apps," https://developers.facebook.com/docs/facebook-login/overview.

[8] U. D. of Health & Human Services, "User-centered design basics," https://www.usability.gov/what-and-why/user-centered-design.html.

[9] C. Bowles, "Looking beyond user-centered design," http://alistapart.com/column/looking-beyond-user-centered-design.

[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.

[13] Git, "Git–local-branching-on-the-cheap," https://git-scm.com/.

[14] K. Padia, "Top 5 open source version control systems," https://jaxenter.com/top-5-open-source-version-control-systems-pros-cons-129538.html.

[15] SVN, "Subversion," https://subversion.apache.org/.

[16] Microsoft, "Platform for intelligent applications," https://www.microsoft.com/en-us/sql-server/.

[17] MongoDB, "Mongodb — for giant ideas," https://www.mongodb.com/.

[18] MySQL, "Mysql," http://www.mysql.com/.

[19] I. B. Network, "Why are we excited to talk about mongodb?" http://www.ibmbpnetwork.com/linux-blog/excited-about-mongodb.

1 WHAT TECHNOLOGIES CHANGED?

Throughout development of the Many Voices Publishing Platform, only a few changes were made throughout development that strayed from the original technology review.

	Technology	Description of What Happened	Comments
1	User Interface Tools	The use of this technology will aid in the development of the user interface, allowing for a beautiful and scalable user interface will help users interact with the platform more easily	Originally the development team decided to use Angular2 as the JavaScript framework and early into development the development team switched to Aurelia as the JavaScript framework to develop the user interface with.
2	User Login & Authentication	User Login and Authentication is important for the application and using a quick implementation of a login system with PHP and SQL to verify account information would be quick to implement.	The development team decided to instead use Google OAuth for user login and authentication as it would allow the heavy lifting to be performed by Google and prevent the development team from worrying about password security.
3	Interface Design	The design philosophy chosen for the application is important for development and the development team decided to use User Centered Design because users are very important for this application	The development team used User Centered Design throughout development and worked towards integrating user feedback into the development process. User feedback sessions were organized with friends and family, as well as minorly with the client Dr. Carlos Jensen.
4	Testing	Testing of application procedures and functions is important because ensuring proper functionality is needed for large applications and performing quality assurance is important.	The development team wanted to integrate Jasmine for integration and application testing ended up using a combination of Jasmine and Travis Continuous Integration.
5	Revision Control Software	Revision Control Software is an important part of any software project and any technology that is used to control revision history is better than nothing.	The development team used Git for all revision control purposes as each member was at least somewhat familiar with the technology.

6	Database	Having an understandable, well built database can aid in the flow of building an application as well as delivery	During the original technology review we decided to use SQL Server from Microsoft. MongoDB ended up being our database system as it is OpenSource and easy to implement.
7	Server Backend	The server is an important aspect of any application. If Ward Cunningham's Federated Wiki is used for the project base, the development team would be using NodeJS	The development team decided to use NodeJS as it has an expansive ecosystem, and allows us to use a common language between the front and backend systems.
8	Text Formatting	Deciding which text formatting material is another important aspect of the project	The development team originally decided to use HTML as formatting for our scraps. Through development we decided to use raw text and enable the use of LaTeX commands within that raw text to provide the functionality that we need.
9	Password Storage	Password storage is a very important aspect of user accounts, and often is one that is brought up in news for the wrong reasons.	Early into development the team decided to put off user accounts till a good portion of the application was completed. The team decided to use Google OAuth instead of using any off the shelf encryption or password storage solutions, negating the use of this technology.

CS Capstone Final Report Blog Posts

June 11, 2017

Many Voices Platform

Prepared for

Oregon State University

Carlos Jensen

Prepared by

Group 61

Remix

Josh Matteson
Steven Powers
Evan Tschuy

Abstract

The culmination of blog posts created during the development of the Many Voices Publishing Platform by Team Remix; Josh Matteson, Steven Powers, and Evan Tschuy for Dr. Carlos Jensen.

CONTENTS

1 FALL

1.1 Week 3

1.1.1 Steven

This week was crazy! Getting our problem statement ironed out took a little longer than we anticipated. Mainly getting my teammates to be able to commit to working on the documents and revisions so we could meet the assignment criteria. Josh was able to meet with me on the 14th. Josh and I continued with my draft of the problem statement and the two of us created the blog platform. It was stressful, but we were able to get everything signed and turned in on time, thankfully.

Last week we met with our client for the first time and layed the plans for the MVP platform as a whole, including long term goals of actually processing the formatting of documents.

For next week I am not sure what to expect, I do know that at the end of the month we have our project requirements documents coming up.

1.1.2 Evan

I definitely had a major lack of time this week. I worked a little bit on some of our documents, like parts of the draft problem statement, but unfortunately was unable to meet in person with my teammates except for small amounts after classes. This week, we start TA meetings and, since the crazy orchestra rehearsal schedules are over, I should have a lot more time in the evening for homework, which frees up daytime hours for any necessary meetings with our client Carlos and with the group.

1.1.3 Josh

Getting Everything finalized was quite a stretch this week, me and Steven had to power through getting our project webpage set up. We also had to get our Problem Statement signed by everybody in the group as well as our client. Next week, we are planning on figuring out a time for our team to meet

consistently, as well as figure out user stories. We ran into some problems this week just trying to get everybody to sign our Problem Statement on time. Another problem we ran into was getting a hold of our client on time. Thankfully, we got everything finished and turned in on time. We'll hopefully be more on top of things next week.

1.2 Week 4

1.2.1 Steven

Trying to get a head in terms of our class work has proved difficult. We haven't heard from our client, but he is busy, so that is understandable. We are attempting to get his feedback on our problem statement as well as involve him in the requirements document creation.

Last week we completed the first submission of our problem statement document but we need to revise it for the coming week. We still don't have specific feedback but we are working on revising it anyways.

For next week I am not sure what to expect, hopefully we are able to involve our client in the documentation creation process and continue as a cohesive group and complete these milestones on time.

1.2.2 Evan

Client communication was a big theme for us this week, with his feedback necessary for us to move forward on our problem statement and helpful for our requirements docs. We were able to meet with our TA and work out a timeslot for that, which is awesome.

For this coming week, the revised problem statement and client req documents are the most important things in the pipeline. Hopefully we'll be able to get client feedback and get those docs done!

1.2.3 Josh

This week we mostly worked on figuring out a time that was going to work for all of us. I think we finally have a solidified time. We also figured out quite a lot of ideas about the tools we could use to design the online application.

Next week, we need to revise our problem statement as well as start working on our Client Requirements Document. Overall, we weren't able to accomplish much this week, but that had a lot to do with the fact that our client hasn't really been responding to us. We now have a real wiki which is awesome.

1.3 Week 5

1.3.1 Steven

This week we were tasked with revising and turning in our problem statement. This was worked on by Josh and I and turned in by Evan. We also began work on the requirements document which has been intimidating! There has been so much stuff to cover.

Next week we have to complete a final draft of our requirements document and then have our client review it and sign it.

1.3.2 Evan

This week we got the revised problem statement signed and turned in. We began work on the requirements, which is a massive effort! I split out a bunch of user stories while we were working on the requirements doc together, and we'll be putting those into the document the following year.

This following week, we'll be getting the final requirements document done and turned in. This week is where I've been pulling more of my own weight, though I still have a bit of a ways to go.

1.3.3 Josh

This week, we managed to get ahold of our client and get him to sign our revised Problem Statement. We previously were having quite a bit of trouble getting ahold of him because he's been traveling. We also managed to all work together on our Requirements Document, which has been one of the few times we've all been able to work find times in our schedule to do it together (even though it was over the internet).

Next week we need to get our client to sign our Requirements Document, which hopefully won't be to much of an issue. Thankfully this week has flown a little more smoothly in terms of our team working together, and hopefully it can continue this way for the rest of the term.

1.4 Week 6

1.4.1 Steven

This week: I made lots revisions to the requirements document, and then integrated the changes that Evan and Josh made to the document. I also attended Dr. Winters writing session and asked about documentation formatting and the like. I also revised our Problem Statement and heard from Dr. Winters that we can resubmit documents once they have been corrected to receive a regrading. This

will hopefully prevent us from receiving another 82/100. I also sent two drafts to our client, with 48 hours notice each time, but neither document has been opened as of this time.

Problems: Our requirements document has slight differences in the formatting compared to the IEEE 830 format specification, though our TA Jon Dodge and Dr. Winters feel that the document looks great.

I emailed our client a draft on Monday, and received no response by Wednesday.

I emailed our client a draft on Wednesday, and as of this time have not received a response nor has our client opened the Requirements document via DocuSign. This is greatly concerning, but also understandable because he is traveling right now.

Next Week: I am hoping that we can get an extension for the signature of our requirements document, as at this time I do not feel that we will have a signature from our client.

I need to speak with my team about contacting our client for another meeting to try and get communication ironed out to prevent these large gaps in communication.

1.4.2 Evan

This week, we did finished requirements document. After meeting with our TA, we polished it and removed sections where specific wording could lead to unnecessary constraints, and we sent it off to our client with the required 48 hours notice. He luckily did get it back, just barely in time for turn-in.

In terms of problems, the only main problem I currently see is client communication. It could be going a little smoother in terms of feedback time, as currently it's mostly send -¿ wait 48 hours -¿ get a signature just in time.

This coming week, I'm heading to the bay area via a camping roadtrip, so I'll not be online as much as would be good. My hope is that I can get enough done early in the week to compensate for the trip; luckily, I'll be back online and able to work by Friday evening giving my plenty of time before the Monday deadline.

1.4.3 Josh

This week was difficult because I had rather large assignments being due at midnight, on nights where we needed to revise our Requirements Document and get it emailed to our client. We managed to finish the revising of the Requirements Document and get it signed just a couple of hours before it was due.

Our Client hasn't been very responsive to our emails, and we're running into quite a bit of trouble just trying to communicate with him in general. We've had quite a few questions about the platform,

but haven't been able to get the proper feedback we need from our client. Hopefully this will change as we continue into the nitty-gritty parts of the project.

This next week shouldn't be too much of a problem, as we've gotten most of the big documents out of the way. Unfortunately, we still haven't figured out a time for all of us to meet weekly. I want to bring this up next class meeting, because us meeting once a week for a fixed amount of time is important. My teammates really pulled through this week because I was so busy with other schoolwork, so shout out to them.

1.5 Week 7

1.5.1 Steven

This week: This week I began working on the Technology Review for our project. I still have more work to do however, as I have a lot more to do on this. I also made major revisions to our requirements document, for better document performance and stability. I made changes that were suggested by Jon Dodge and also reworded some portions of the document for better readability.

Next Week: Next week I will continue on the Technology Review for the early portion of the week and continue to revise and update our past documents. I will also contact our client again about a possible meeting for a few points of clarification.

Problems: Adding bibliography entries for the requirements document proved way more difficult than I anticipated. Having previous experience with BibTeX, I assumed it would be smooth sailing. It turns out I was using square brackets instead of curly braces as the primary identifier. Once I fixed that, I had to trace a few other errors. Additionally I didn't add the Abstract to our document, so that needs to occur as well.

1.6 Week 8

1.6.1 Steven

This week: This week I continued working on the Technology Review for our project. I made some formatting changes to make our document cleaner, within the Latex document and the appearance on the PDF. We also began planning on the design document and what that will entail.

Next Week: Next week I will continue on the Design Document, we still have a few concerns about our technologies, as to be expected, as we haven't started on the implementation portion yet. I will also contact our client again about a possible meeting for a few points of clarification.

Problems: Keeping in touch with our client has been difficult. With our client traveling or otherwise unavailable has let us slip out of communication. We need to do better on this front.

1.6.2 Evan

This week, I worked with the rest of the group to finish the tech review. Afterwards, Steven made some changes to make it a little better, so shoutout to him! Next week is thanksgiving, so I might try to get some things done on the design doc while up with family in Portland. Our client hasn't been the most available, so we're pushing forward with the information available to us.

1.6.3 Josh

This week we cracked down on our Technology Review, and managed to get some extra time to compensate for other homework due those days. I still haven't learned LaTeX as well as I want to, but under the circumstances, I think it went okay. Steven managed to go over some formatting errors that happened for our tables, which made things look quite a bit better.

The following week is Thanksgiving week, so we won't have very much time on campus to be able to get everything done. This means that we need to crack down when we get back after the short break. We're going to start working more on implementation.

We're still running into problems with contacting our client. I wasn't able to do a whole section on the technology review because I needed more clarification from him, but we haven't been able to get a meeting with him in since the first month. Another problem was not being able to use LaTeX that well, but hopefully it'll go smoother in the future as I've learned a bit about it this week.

2 WINTER

2.1 Week 1

2.1.1 Steven

2.1.1.1 PROGRESS: For this week we were getting back into the groove of things after Winter break. Josh and I were able to meet with Carlos the week prior and talked about arranging meetings going forward each week.

2.1.1.2 PROBLEMS: We weren't able to get a whole lot of work done this week, but we planned for meeting each week at least a few hours a couple days of the week so we can continue making progress.

2.1.1.3 PLAN: We plan to meet next week and work on the framework and user stories.

2.1.2 Evan

This week, we're not really working on too much. We'll be meeting with Carlos in about a week and a half, which will hopefully give us an opportunity to solidify what he wants, versus what we've been designing.

In the mean time, I worked on some of the book management backend over break. I'll continue working on that in my free time.

2.1.3 Josh

The very end of the break, we were able to get a meeting with Carlos to clear up some possible fallout from the previous term. While we were able to get confirmation on some of the technologies we are using, we didn't sort out a few in the backend. We were able to, however, figure when we could meet with him the following weeks after he moved into his new office. This means we won't be meeting with him for at least two weeks.

As far as figuring out user stories and possible work we could be doing, we managed to develop a game plan of how we can divide and conquer most of it.

2.2 Week 2

2.2.1 Steven

2.2.1.1 PROBLEMS: This week we began our weekly meetings, though our client had to cancel meeting for the first two weeks.

2.2.1.2 PROGRESS: We also setup our meetings with Jon, which was actually pretty difficult to find a time that worked for all four of our schedules.

2.2.1.3 PLAN: We plan to continue working on the User Stories and Framework so we are able to make progress towards a working prototype.

2.2.2 Evan

This week we set up meetings with Jon and I've fleshed out the required API for the backend. Barring issues, I'm going to be implementing that for the next few weeks.

2.2.3 Josh

Our group managed to meet this week and accomplish setting up the basic architecture of the application. Me and Steven are moving forward with Aurelia and are doing are best to get it configured.

2.3 Week 3

2.3.1 Steven

2.3.1.1 PROBLEMS: This week we were to begin our weekly meetings with our client. Our client recently transferred positions and our weekly appointment was discarded. This resulted in our weekly time slot being given away, so now we have to find a new time (planned for 9:00AM Mondays). Our client also wants to only meet every other week rather than weekly now...

2.3.1.2 PROGRESS: We talked over the past terms documents and possible upcoming documents that we would be seeing in the class and that we should be focusing on development throughout this term.

We worked on the backend system and testing of the include and input commands with our setup (which seems to be working well!). We also looked into security practices to protect our LaTeX documents from succumbing to common exploits (escaping the document and accessing the shell).

2.3.1.3 PLAN: We also worked on getting the framework up and running (TS Lint has been giving us some issues, so we might go with straight typescript without LINT and go to JavaScript with Lint.

2.3.2 Evan

This week Carlos cancelled due to calendar migration issues, so we've rescheduled for next week at 9am. Hopefully that'll work out...

I'm working on getting that backend still moving forward.

2.3.3 Josh

The major changes this week had to do with our client switching his availability to once every two weeks, this hasn't been too much of a problem yet.

The backend has been progressing with Evan, and me and Steven got a semi working version of the Javascript framework working for Aurelia. Me and Even were able to meet this week briefly to discuss progress.

2.4 Week 4

2.4.1 Steven

2.4.1.1 PROBLEMS: No Specific Problems This Week.

2.4.1.2 PROGRESS: This week we met with Dr. Jensen and discussed the back-end of the system as well as some plans for how to model the front end of the system.

Dr. Jensen suggested using low fidelity / medium fidelity to receive feedback from users instead of focusing on getting to high fidelity and having users review that as they will be less critical of the whole system and instead focus on pixel alignment.

2.4.1.3 PLAN: Josh and I need to get more work done on Aurelia and get a basic user interface up and running and try working towards in page rendering of the resulting PDF from the back end system.

2.4.2 Evan

This week we got to meet with Dr Jensen about the state of the backend system (which is my primary domain) and the front-end design, which Steven is planning on drawing some rough prototypes for.

My plan for the next few days is to work on more backend integration, including getting the textbook to render properly.

2.4.3 Josh

We finally got the ball rolling this week because we were able to meet with Dr. Jensen. This led to discussion about work flow designs with a low fidelity goal.

Me and Steven got a fully working version of Aurelia working on a local server, and now we're working on the the basic layout.

Evan has still been making good progress on the backend, and was able to show Dr. Jensen the basic foundation of the backend.

2.5 Week 5

2.5.1 Steven

2.5.1.1 PROGRESS: This week was a week we had off from meeting with Dr. Jensen, we used this time to work on developing some of the required features that we need. This included Josh and I working on the PDF in browser rendering with Aurelia. Josh finished this task up separately.

I worked on Prototype development to have Dr. Jensen review and provide feedback on during next weeks meeting.

2.5.1.2 PROBLEMS: I wasn't able to get as many prototypes drawn as I would have liked, but there is still a good chunk of design that Dr. Jensen could critique.

2.5.1.3 PLAN: Next week I plan to show Dr. Jensen our designs and take in any feedback he might have. I also plan to work on revising all of our documents and get them placed into our OneNote document that I have prepared for our group.

2.5.2 Evan

We didn't meet with Carlos this week, as we've made those every-other. Instead, we used the time to polish existing features we'd already worked on, and write new frontend and backend features. Personally, I got textbook rendering working, which was annoyingly difficult, but it's finally done!

At our next meeting, we're primarily going to show Carlos our frontend prototype designs to get feedback on user flow, interface direction, and the like.

2.5.3 Josh

Me and Steven started working on a PDF viewer by following a guide online. This led to some good progress as far as knowledge for the two of us. We have our meeting with Dr. Jensen this week, and we want to have this done before showing him.

Steven worked on prototypes and I branched off into finishing the pdfviewer demo.

2.6 Week 6

2.6.1 Steven

2.6.1.1 PROGRESS: This week we met with Dr. Jensen and he had lots of feedback about our designs, mainly in ways to simplify the interface for the users while also making it easier to develop. We talked about rendering the PDF only when needed (as in active editing) instead of constantly. We also talked about the possibility of using a tabbed system to only render when the user wants to, which will reduce our server calls.

2.6.1.2 PLAN: I plan to continue working on our document revisions, as well as our Progress Report materials (video, presentation materials, and report), which will mainly summarize the previous terms work and provide an overview of our progress this term.

2.6.1.3 PROBLEMS: Something that will be better next week is availability, as this week has a lot of midterms for various classes.

2.6.2 Evan

After our meeting with Carlos, we were able to refine our vision for the interface. He mainly stressed making it simpler, and making it less duplicative. We now have an idea around how the user will go about editing a scrap, editing a chapter, etc., and when to render PDFs.

This week is mainly going to be capstone assignments – the revised documents and the video summary.

2.6.3 Josh

Our meeting with Dr. Jensen went very well, he gave us constructive feedback on our designs and we were able to demo the front of the application to him. Me and Steven are going to continue working on the pdfviewer and integrating it into the application.

We also need to work on a progress report, which will be in the form of a video with edits from last time. We're going to demo our front end as well as our back end for this. It's due on Friday, and with Valentine's Day in the middle, it might be difficult to accomplish everything without the use of an all nighter.

2.7 Week 7

2.7.1 Steven

2.7.1.1 PROGRESS: We didn't meet with Dr. Jensen this week and instead spent some time working on our user stories and projects for going forward.

2.7.1.2 PROBLEMS: Josh and I need to get together more so we can work through a lot of the front end user projects and make some headway on the design so any minor tweaks to the prototypes will result in minor changes to the design.

2.7.1.3 PLAN: I do feel we are a little behind, but next week we are planning on going through API calls from the frontend to the backend and vice versa to make this planning stage a little better.

2.7.2 Evan

After not meeting with Dr Jensen this week, we were able to hammer out user stories for the future. We have a lot to work on but honestly at this point I feel comfortable we will be able to get all of this done on time.

2.7.3 Josh

No meeting with Dr. Jensen this week, but we were able to hammer out all our user stories and figure out what we need for both the front end and the backend. As far as the front end goes, I feel like me and Steven still have quite a lot to accomplish, but hopefully we'll jump on that soon.

2.8 Week 8

2.8.1 Steven

2.8.1.1 PROGRESS: This week we went through all of our anticipated API calls from the frontend to the backend systems. Having these API calls is important, as it allows for Evan to implement the functionality we need for the frontend while Josh and I make progress on the frontend goals.

2.8.1.2 PLAN: Josh and I plan on getting together to work on implementing the Aurelia router, which will allow us to have a single page application.

2.8.1.3 PROGRESS: Our client is happy with the progress we have been making as well. We ran through some user testing with Carlos and he had lots of feedback as we continue to iterate on our designs.

2.8.1.4 PLAN: This is something I plan to work on through the rest of the term so we can continue our user testing and integrate their feedback.

2.8.1.5 PROBLEMS: No Specific Problems This Week.

2.8.2 Evan

This week I made sure everyone made it to our group meeting on time, as it was necessary for all of us to go over API calls for me to build on the backend. This is the contract between me and the frontend that lets me know exactly what to write. Carlos is happy with our progress, and after going over Steven's mock interfaces with him, we feel good about the direction the frontend and backend are going.

2.8.3 Josh

As far as our meeting this week, we had good success with all we got done for Dr. Jensen. We showed all our layout prototypes, and he seemed happy with the results.

This week we were able to meet up and completely go over the API calls and backend requirements, which set us up really nice moving forward. Because of this meeting, me and Steven now understand what we need to accomplish and make available as soon as we get the chance.

As far as the progress Steven and I have made on the frontend, we are planning to meet together this week to work on implementing a router (enables us to switch between pages without refreshing).

2.9 Week 9

2.9.1 Steven

2.9.1.1 PROGRESS: This week consisted of Josh and I spending time together and working through getting the Aurelia router integrated. We were able to do so, but we had a few issues with getting the application to work on another machine besides Josh's computer. We plan to work through this during our team meeting, as having the application only work on one computer is not ideal.

2.9.1.2 PROBLEMS: Josh and I also worked on Aurelia tabs, for tabbed browsing on our one page application, but ran into some issues getting everything working.

2.9.1.3 PLAN: We plan to continue working on this next week and hope to have it working by finals week. We have a meeting with our client next week and will talk about the Aurelia router and Aurelia tabs.

2.9.2 Evan

Unfortunately, I was fairly busy this week. I wasn't able to get a lot done, but was able to meet with Steven and Josh while they were getting the frontend set up to work on both of their machines. I was able to help debug a little.

2.9.3 Josh

Steven and I were finally able to get a good amount of work in today covering the front end. We worked on implementing the router and trying to get basic tab functionality working, unfortunately we weren't able to the tabs working. This is a big accomplishment for us non-the-less. Carlos asked me to start thinking critically about an algorithm for recompiling latex into a PDF, so I'm going to start thinking and doing research on that topic before we meet again.

2.10 Week 10

2.10.1 Steven

2.10.1.1 PROGRESS: For the tenth week of the term, I worked with Evan and Josh to get the Aurelia project running on my Hackintosh machine so we can continue to develop now that we are using the one page application, which requires the Aurelia project to be running. Before this I was able to make changes and simply refresh the page. This has been a minor inconvenience, but the router is worth it.

2.10.1.2 PLAN: For our meeting with Carlos last week, we decided to skip a meeting next week, as we all would be busy with Finals, himself included. We plan to start again first week of Spring term.

Next week I don't envision having a lot of time to dedicate to the project beyond documentation for the progress report and the video. I plan to work through Spring break however so we can continue to make progress.

2.10.1.3 PROBLEMS: No Specific Problems This Week.

2.10.2 Evan

Finally the term is coming to a close! We've decided to put off our next meeting to the first week of spring break. At this point we're overloaded with other projects and don't anticipate having time to work on the technical part of the project, but we will be able to get together to build the final presentation done soon.

2.10.3 Josh

This is the last week of the term, and unfortunately we weren't really able to accomplish more from last week because of finals coming to a close. We also decided not to meet with Carlos this week in order to give us more time to bring something deliverable to the next meeting. Spring break is finally here! (sorta)

2.11 Week 11

2.11.1 Steven

2.11.1.1 PROGRESS: For finals week we did not meet as a group for application development because of other classes and studying getting in the way. We have communicated as a team about our individual progress however. We were able to come together to work on our progress report video together so we could get it turned in on time.

2.11.1.2 PLAN: Continue working on the application during spring break.

2.11.1.3 PROBLEMS: No Specific Problems This Week.

2.11.2 Evan

This week has been mostly the progress report. We have successfully finished the videos and I managed to finish my progress report this afternoon. Can't wait til next term!

2.11.3 Josh

3 SPRING

3.1 Week 1

3.1.1 Steven

3.1.1.1 PROGRESS: For the first week of the term I made a lot of progress on the frontend development, changing our current implementation to match more of our prototypes: vertical menu layout as well as page skeletons for each of our router pages. I also made modifications to our PDF viewer, as it was appearing much too zoomed in for my liking. I am also working on making additional page changes and building our bare-bones scrap editor. Josh and I will need to work with Evan to join the backend to the frontend.

3.1.1.2 PROBLEMS: During Spring break, Josh's laptop was stolen from his home. Thankfully it has since been recovered, but it has put us behind just a little because we are having to get his system up and running again. Thankfully we had a recent commit from his machine so we didn't lose too much completed work. This did however put a damper on getting together over break to make big leaps and bounds during this 'free' time, as Josh had to get in contact with the police that had recovered his laptop. Josh hopes to recover his laptop from the police impound this week.

3.1.1.3 PLAN: Next week we plan to work on our Page Tabs implementation again, as it is important to our design. I plan to finish up working on the page skeletons and scrap editor page if I am not able to do so with the remaining time of this week. Additionally next week will consist of working on the various page views.

3.1.2 Evan

3.1.2.1 PROGRESS: This week I sat down with Steven and Josh to hammer out exact dates we want specific functionality finished. This will give me deadlines to aim for over the next month. For this first week of the term, my goal was to go from a tiny Node script that could respond "hello world" to a multi-file, properly set up Node application that wraps the already-written library I worked on over winter term. It's currently possible to do things like create a book, edit a book [metadata or adding chapters by id], get the book's history, and generate a PDF of the book.

3.1.2.2 PROBLEMS: Josh's laptop got stolen! Obviously having one member of our team out of commission during April is not ideal. He's working on getting that resolved, though, and I'm confident the frontend team will be able to get back on track. I've offered to help anywhere needed.

Personally, this past week has actually been one of the smoothest sailing I've had in a while for this project.

3.1.2.3 PLAN: Next up is the things required for the next due date of the frontend: by April 10, I plan to support the creation/update/history/PDF generation of scrap objects. I'll also need to get the ability to list a user's books.

3.1.3 Josh

3.1.3.1 PROGRESS: Not a considerable amount of work was able to be done on my part this week, and unfortunately it didn't go to well in the client meeting. The rest of my group was able get some good work done on the backend and front end.

3.1.3.2 PROBLEMS: My laptop got stolen over spring break, and that drastically set me back. I've been working to get it back and up and running again. I will have a lot to catch up on when I finally get it working again, as my laptop had the correct working environment.

3.1.3.3 PLAN: The plan for the rest of this week is to get everything working and discuss with my team what we can be planning and getting ready for when I can use my laptop again. I think everything should go a little more smooth from then on. Our plan is to write out every task from now until it's completely finished.

3.2 Week 2

3.2.1 Steven

3.2.1.1 Progress: This week Josh and I made a lot of progress on our development and were able to check off a lot of our design goals from our Trello front end project management dashboard. These include some of the different collection pages. Additionally we are close on our implementation of the scrap editor, though sending to the backend has not been achieved.

3.2.1.2 Problems: User authentication and login is currently not setup and instead we are currently providing the user in the URL to specify which user information to retrieve. Currently we are using postman for hosting our "database" so we will need to fix that at a future date.

3.2.1.3 Plans: Josh and I plan to get together this weekend and continue to cross off different tasks together, and next week continue working together as a team and prepare our poster draft two.

3.2.2 Evan

3.2.2.1 Progress: This week we made a lot of progress on the backend and frontend! I've written up entire spec documents for detailing POST and GET endpoint routes, return values, POST

bodies, etc. I've made it possible to do anything that is on the frontend trello before their due date.

3.2.2.2 Problems: I have yet to get authentication working. It's on my to-do list at top priority, but it involves setting up a MongoDB instance to manage the conversion between Google oAuth tokens and our own login tokens. I'm not worried about it, though, and I'm sure I'll be able to get that done soon.

3.2.2.3 Plans: Currently, my TODO has two main parts:

1) version pinning

2) things requiring a MongoDB instance. That's things like favorites, login, unassociated items, etc. Those are on the plate for next week. In the week after that, my plan is to do search.

3.2.3 Josh

3.2.3.1 Progress: Thankfully, this week was much more productive that previous weeks. I had my laptop and Steven and I were able to make make progress on our goals and user stories. These include some of the different collection pages. Right now we're working on finishing the skeleton for the pages.

3.2.3.2 Problems: We don't have a real data base that we're working with as of yet, because we're still experimenting with backend calls. We don't have any real talk from the frontend to the backend.

3.2.3.3 Plans: This weekend, Steven and I will be working through more of our frontend stuff. It's looking like Friday's will be major work days for the three of us. This is much needed as we have a lot do to in the next couple of days.

3.3 Week 3

3.3.1 Steven

3.3.1.1 PROGRESS: This week we made a lot of progress in terms of raw development. Evan was able to put some fresh eyes onto the parameter passing issue, so we were able to get the my books pages into a much better position. Currently I have a semi-working edit chapter page. I also made a lot of progress with the my chapters page, so now you can preview a given chapter from a book. I am working on bringing the same progress to the my scraps page, so we can preview a scrap.

3.3.1.2 PROBLEMS: This week our client scheduled over our meeting time so we were unable to meet with him. We planned to show our poster draft and seek changes and / or approval. Editchapter is having some issues at the moment, because of how we are passing around the parameters.

3.3.1.3 PLAN: We plan to attempt to meet with our client next week, we are worried about this because there is only a few days before the deadline, as the deadline has been moved forward a few days.

3.3.2 Evan

3.3.2.1 PROGRESS: This week I successfully integrated authentication into the application using a Mongo backend. This required the entire Mongo setup along with it, which was a relatively major undertaking. It is now set up on our staging server, though, which is good news. This will allow me to much more quickly write favoriting and unassociated document tracking, both of which are going to use the Mongo database.

I also have search now working. Every object update sends an updated copy of itself to an instance of Elasticsearch, which then can be queried to find books and chapters based on their names, and scraps based on their text.

3.3.2.2 PROBLEMS: Currently, everything seems smooth sailing. The main issue at this point is the amount of time that we have left to complete the project, but so far, if I keep my pace, I know the backend will get done on time and that if Steven and Josh keep their pace, we'll have a frontend that can be shown off with it.

3.3.2.3 PLAN: Currently I have the documents stored in Elasticsearch but need to expose a search API that the frontend can use. That's on the plate for this weekend, along with finishing favoriting and perhaps unassociated objects.

This will leave me with:

1) version pinning 2) forking

3.3.3 Josh

3.3.3.1 PROGRESS: This past week has had substantial work done for everyone in the group, me and Steven have been working a lot with routes and tabs, as well as the layout of the page more. A major roadblock that we had was not getting one of our sub-routes to work properly with passing an argument to it. Evan was then able to help us with it and managed to fix the problem. Now we're able to finish the rest of the editors and work towards finishing the project.

3.3.3.2 PROBLEMS: As mentioned earlier, Steven and I got caught up on managing the sub-routes with a parameter. This took us both a lot of time which meant we couldn't really focus our time on anything else. Another problem that we ran into was our client rescheduling on us and we couldn't get feedback and other things done because of that.

3.3.3.3 PLAN: This next week we all plan to meet A LOT in order to accomplish all our goals. Being as the code freeze is in little over a week, we have a lot to get done first before then. We will finishing all of our editors, integrate a drag and drop feature, and a few other things.

3.4 Week 4

3.4.1 Steven

3.4.1.1 PROBLEMS: This week we have been having problems with simply having enough time to get everything done. Having spent so much time on capstone these past few weeks, while making good progress towards our goals, has put other classes on the back-burner.

3.4.1.2 PROGRESS: Last week and especially this week, we have made a ton of progress now that our blockers have been dealt with. Having learned how Aurelia handles routes and child routes differently has unlocked our ability to develop the more extreme features of our application. We are in the final stretches for being feature complete, and once that is finished, we plan to focus on bug fixes and testing of our application from top to bottom.

3.4.1.3 PLAN: If we have additional time after these two tasks, we plan to add more functionality in terms of stretch goals and anything else we can think of before the code freeze. After the code freeze we plan on continuing to work on the application for our client.

3.4.2 Evan

3.4.2.1 PROGRESS: We did it! Almost. We've made massive progress, and I think by Monday we will have a deliverable to be... proud isn't the right word, but acceptable! I worked hard on getting things like Edit Chapter working, integrating with Search, de-duplication of Search pages, etc. I also integrating logins, so now users have a fancy welcome page and login button before they can access their content. I also got drag-and-drop working, which is super fancy.

3.4.2.2 PROBLEMS: We have three days left. Yikes. I've been doing mostly entirely capstone, so I'm now full up with other homework to do, which is now weighing down. I'm happy with how much I've done at this point, though, and I'm confident that I've given my teammates enough of a boost that they're fine with me spending this time on other things. I've still got several tasks on our issue tracker, which I'm still planning on getting done, so I'm feeling good.

3.4.2.3 PLANS: My main issue left on our Trello is favoriting: how do users add a favorite? Where do they see their favorites? That kind of thing. I also have a few backend issues to resolve if I have the time. Other than that, it's mostly just pushing ahead to our deadline.

3.4.3 Josh

3.4.3.1 PROGRESS: This week, the team made major progress with accomplishing the user stories that we set out. I mainly focused on the search engine on the frontend, which Evan had already set up on the backend. The rest of my team has been working consistently on the application editors, as well as the CSS and login. I haven't been a little behind because of other classes needing attention, but thankfully Steven and Evan have been working thoroughly regardless. We met with Dr. Jensen on Tuesday and were able to get our poster signed as well as show him the incredible progress we've made on the app. He seemed very pleased, although he did mention a few improvements that we've since been working on.

3.4.3.2 PROBLEMS: As mentioned above, I've been swamped with other homework. I'm trying to work as much as possible on everything so I can have the weekend to fully focus on senior capstone with the rest of my group.

3.4.3.3 PLAN: This next week I plan to integrate unit test into our application. After this I plan to make a majority of the test for our application. I will also be helping wherever possible on other parts of the application.

3.5 Week 5

3.5.1 Steven

3.5.1.1 Progress: We have made it through most of development and are feature complete. We plan to continue making tweaks and improvements where we can going forward however. Evan and I were able to get the remaining issues completed while Josh had to deal with other classes / events.

3.5.1.2 Problems: We have again lost our scheduled meeting time through no fault of our own. So instead of meeting Mondays at 1:00PM, which worked for everyone on the team, we now have to find ANOTHER new time that works for all of us, as well as Carlos... In terms of the project, we had some problems mainly with finding the time to work on the project.

3.5.1.3 Plan: I plan to continue working on the Wired reviews for Friday. I interviewed Courtney Bonn of the Calvary Corvallis iOS and Android applications about her project. After that I plan to work on our Midterm progress report and video.

3.5.2 Evan

3.5.2.1 Progress: We're done! Finally done with the programming portion. There are perhaps a few things to cleanup on the frontend but we're more or less done. The work was finished up mostly by Steven and I and we managed to all meet on Monday to finish up straggling issues.

3.5.2.2 Problems: At this point, my ”problems” are moreso doing all of the assignments that got put off for capstone. There’s also the WIRED doc and the midterm video we need to do, so I’m working on my WIRED thing due today. We’re considering what needs tweaking on our site between now and expo, but it shouldn’t be too much.

3.5.2.3 Plan: Right now, it’s mostly sitting tight until we get assignment specifications for the video. Other than that, we’re pretty much just doing our individual WIRED articles, so nothing group work wise.

3.5.3 Josh

3.5.3.1 Progress: This week, the team managed to get all of our required features completed. I was assigned to getting a working timeline implemented, and making scraps able to upload images. I was also tasked with implementing a pretty view for our application, but I also had a report due that night and so I wasn’t able to get the last task done. The other two I was able to complete fully however.

3.5.3.2 Problems: We are having continued problems with scheduling time to meet with Carlos, as everything we have scheduled lately has fallen through on his part. Last weekend, I was unfortunately caught in the middle of other important obligations that landed on the most important weekend of our senior capstone. I was able to get most of the requirements assignment to me done, but I didn’t struggle a little with the last task and getting my other homework done. Thankfully my team was able to carry the rest of the responsibilities.

3.5.3.3 Plan: Now all we have left is preparing for expo and getting our midterm report in on time. We’ll be finalizing tweeks for our client in the mean time.

3.6 Week 6

3.6.1 Steven

3.6.1.1 Progress: This week we did not focus on the application and instead we have been focusing on our report and presentation. Expo is coming up and we have been preparing for this and plans for next week. Our client has found a time that works for him that allows us to make up the meeting after it was cancelled.

3.6.1.2 Problems: We were unable to meet with Carlos last week, and this week due to our meeting time being given away. We are working on scheduling another meeting and getting our progress report and presentation completed.

3.6.1.3 Plan: This weekend and Monday we plan to continue working on our progress report and presentation so it is ready for Monday night.

3.6.2 Evan

3.6.3 Josh

3.6.3.1 Progress: This week, not a whole lot was accomplished in terms of the application. Our team has mainly been focused on getting details about our report and presentation figured out. We didn't meet this week, as we didn't really have much to go over or discuss. I've been working on the report more to make up for my inability to help as much as the others during the final days of the code freeze.

3.6.3.2 Problems: Still haven't been able to meet with Carlos, so hopefully that won't fall through again. No other real problems this week.

3.6.3.3 Plan: The plan as of today is to get as much of the project and presentation done myself until I meet with the rest of my group tomorrow to finalize our presentation and go over everything. Expo is getting closer, so we will also be prepping for that.

3.7 Week 7

3.7.1 Steven

3.7.1.1 Progress: We met with Carlos this week and were able to show all of the progress that we have made over the time between our last meeting. Carlos was very pleased with our progress and only had a few changes to suggest before demonstrating our application at expo.

3.7.1.2 Problems: We have a few changes to make to the application, as well as the implementation of Travis CI (Continuous Integration). The logout button has been causing some issues as well, as we are having trouble activating the logout with the API. Currently we clear the cookies and reload the page.

3.7.1.3 Plan: We will continue working on all changes that we have planned and continue to add additional features where we can.

Josh and Evan plan to take a look at the logout button, so it may be working before expo!

3.7.2 Evan

3.7.2.1 Progress: We finally were able to meet with our capstone sponsor, Dr Carlos Jensen, one last time before Expo. We showed him all of the progress we had made in the significant time

since our previous meeting. He was very happy with what he saw, though he did have a few pain points that he brought up.

3.7.2.2 Problems: There were a few issues that Carlos pointed out that we had to spend some time fixing. In addition, we made a final push to get things like TravisCI, a continuous integration platform, up and running. This allows us to automatically have our tests run every time we push, which allows us to catch issues faster.

3.7.2.3 Plan: Expo on the 19th! And then we're finishing up mostly the required capstone documents. There's a decent chance we'll be finishing some polish for Carlos, too, just for good measure, but nothing too groundbreaking.

3.7.3 Josh

3.7.3.1 Progress: This week, we were finally able to get a meeting with Carlos, this included us showing him all our progress and him having suggestions on what we should implement in order to get ready for expo. The rest of this week has been dedicated to figuring out travisCI for our continuous integration, and preparing for expo. We are finally ready and will hopefully have our application working at 100

3.7.3.2 Problems: We are having trouble with TravisCI, as it only seems to be working on my computer and not everyone else's. So I'll be working on that and hopefully getting it working soon. Steven has been working on getting a logout button for our application, and hopefully we'll have that all up and running.

3.7.3.3 Plan: After expo, we just need to continue working on what we have left. This week is crazy with every other class having due dates, so hopefully we'll all be able to balance our homework and what we have left for this class.

3.8 Week 8

3.8.1 Steven

3.8.1.1 Progress: I enjoyed working on this project, when we weren't stressed about deadlines and documentation. I feel that our project has turned out quite well, especially for the problems we had at the beginning of the year. I feel our application is unique and works well. I walked around Expo and realized that many teams did not complete their projects unfortunately. This made me feel better about my position however. I didn't like going in blind to some of the technologies of our application. This did however help with connecting with my teammates.

I learned a lot of backend technologies while working with Evan, as well as I learned a lot about different JavaScript frameworks like Aurelia while working with both Josh and Evan. Even though they had not worked with this specific framework, they have had experience with other frameworks that work in similar ways. I would say that this is the biggest technical skill I am taking away from this project. I also gained a lot of experience with performing managerial duties and making sure the project was advancing to meet certain deadlines.

I feel that I will me using my skills as a manager, JavaScript framework skills, and web development skills as a whole in the future.

3.8.1.2 Problems: If I was the client for this project I would be satisfied with the work completed, but I wouldn't be satisfied with the communication that took place. I am not satisfied as a student and developer of this project. Communication played a big part of this project and without having an ample supply of it, we were often left wondering how to proceed and discouraged the team from jumping in with both feet.

3.8.1.3 Plan: In the future, if I were to continue working on this project I would feel that a more organized backend that supports scalability at an increased rate would be beneficial. I also feel that if we had more time we would get to performing long term user studies and make modifications to he user interface to make it more intuitive. We throw a lot of options at the user without introducing them slowly. Another good future implementation would be creating a tutorial system that walks the user through the creation of a scrap, chapter, and book, as well as the other features of the application.

If I were to redo this project again, I would tell myself to start working on the larger portions of the application sooner. Having to figure out the big pieces before we could advance to the next tasks resulted in not much getting done at the beginning of development. Once these big pieces were completed, at least enough to continue with development everything else fell into place.

3.8.2 Evan

3.8.2.1 Progress: It has been a crazy year. Going into the project, I was never 100

I learned a lot from my teammates this year. Communicating my exact goals for my portions of the project, and how to fit them in with their portions, was a main issue I had to consciously work on. Speaking of communication, we managed to go from barely meeting fall term to frequent meetings in early spring term. I believe that this was critical for delivering a product that he would like – I know if I were him, I would definitely be smiling upon the result. It delivered a much cleaner and usable platform that I think many groups would have been capable of. I liked the ability to stretch my web

dev muscles and develop the platform as quick as possible nearing the due date.

3.8.2.2 Problems: If I were our client, my main concern would have been communication. In the end, though, we worked that out, and delivered a product to his specifications. I think that it looked rough for a few weeks, but in the end, we delivered a product that I am proud of and that I would be happy accepting if I were the project client. I have no real complaints about the project overall, save the general capstone document avalanche. The avalanche is excessive and was **nothing** but a **huge** timewaste. I learned nothing from doing the documents and would be perfectly happy printing out a copy of every document I worked on and using it on my next camping trip as starter.

3.8.2.3 Plan: In the future, I see myself using the web skills for any of my one-off personal projects. I'm not planning to go into web development, but I also see the software architecture skills potentially being of use in the future. In the future, if our client were to continue the project into the next year, I would most want to see the document storage module overhauled, and perhaps the user interface could undergo user studies and a minor flow overhaul. However, I'm happy with where the project wound up and think we have achieved a good "minimum viable product" release.

3.8.3 Josh

3.8.3.1 Progress: Overall, I really enjoyed certain aspects of working on this project. My group mates were helpful and informative in areas where I wasn't, and we all balanced each other out pretty well. I think if I had to redo this project from fall term, I would start much much earlier on the heavy portions. I feel like I do this with a lot of my projects, but this one specifically could have been done with a lot less stress involved had we gotten a better head start. I think it's something that would have helped out the rest of my classes as well.

By far the biggest skill I've learned over this period of time is my web development skill. I already had a good understanding of how everything worked from a basic point of view, but this project has definitely increased and pushed my knowledge. I think I can look at web applications and understand them from a more advanced level now.

One of the skills I see myself using in the future is handling group assignments and how to delegate work among everybody. Fall term was a big learning process for everyone, and we didn't mean nearly as much as we should have. Thankfully by Spring term we were able to work more progressively.

What I really enjoyed about the project was knowing exactly what we had to do and then just carrying out the plan. It's always frustrating when you're scrambling for something productive to

do and not being able to have everything set up ahead of time. What I wasn't a huge fan of about senior capstone is that the hardest part of it happens in the best time of the year. This also could have probably been prevented had I prepared a little better.

I learned that my teammates have a lot of valuable knowledge in different areas than me, and this ended up being crucial to the success of our application. I learned that my teammates are pretty reliable as well.

3.8.3.2 Problems: I think that if I was my client, I'd be a little shocked at the progress that our team made in a short amount of time at the end. One of the biggest problems we ran into was poor communication between our client and us, and this lead to very little knowledge of how far a long we were with the project. I'm sure that he was frustrated with us at points, but if I were my client, I would be very thrilled with how it turned out.

3.8.3.3 Plan: If I were to continue working on the project through the following year, I would start by doing way way more testing. Our application didn't have nearly as much testing done to it than was necessary. I think this was pretty troublesome when we started finishing the application. After thorough testing, I think I would start focusing on user feedback and implementing that better.

CS CAPSTONE FINAL REPORT FINAL POSTER

JUNE 11, 2017

MANY VOICES PLATFORM

PREPARED FOR

OREGON STATE UNIVERSITY

CARLOS JENSEN

PREPARED BY

GROUP 61

REMIX

JOSH MATTESON
STEVEN POWERS
EVAN TSCHUY

Abstract

The culmination of blog posts created during the development of the Many Voices Publishing Platform by Team Remix; Josh Matteson, Steven Powers, and Evan Tschuy for Dr. Carlos Jensen.

MANY VOICES PUBLISHING PLATFORM

Collaborative Textbook Publishing

Why does it matter?

TEXTBOOKS NEED IMPROVING

Today, textbooks are filled with yearly updates that are often released with the minimum number of changes needed to justify a new edition.

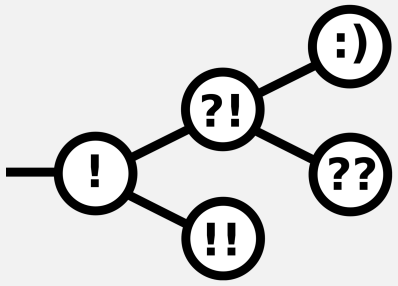
Textbooks

- can cost hundreds of dollars
- are often outdated or contain incorrect information
- take a long time to write and review

HOW COLLABORATIVE TEXTBOOKS HELP

The Many Voices Publishing (MVP) Platform, instead, allows professors to create their own textbooks by using crowd-sourced content.

- This monetary benefit will allow professors to update their courses more frequently without passing on large costs to the students.
- Content creators are able to submit materials of all different formats, including: images, small pieces of text, pages, or even entire chapters.
- A knowledgeable professor can contribute a chapter or unit on their specialty, and others are able to modify obtained materials in the way they see fit.
- Provide the ability to publish their textbook as a PDF when they desire.



The Many Voices Publishing Platform team logo representing the remixing of ideas and content

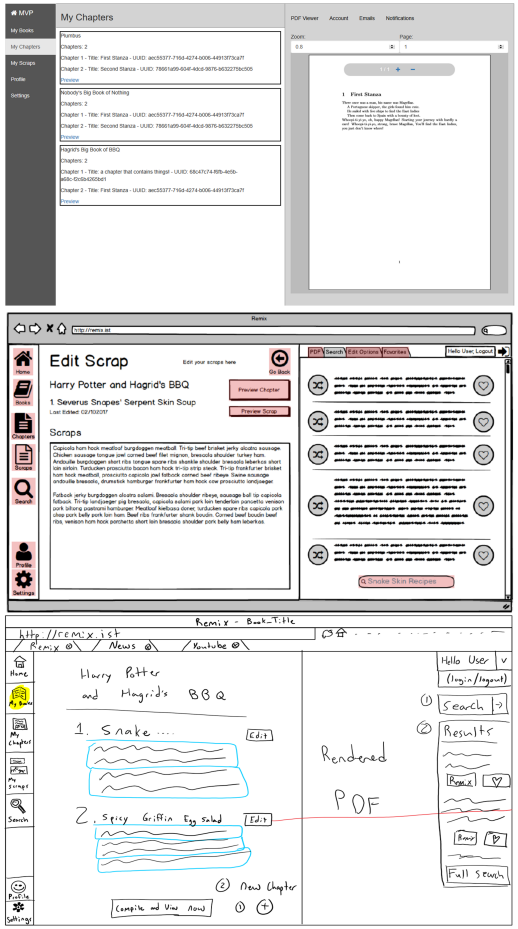
PROJECT OVERVIEW

A remixed textbook created with the Many Voices Publishing (MVP) Platform consists of a textbook object that contains chapters. These chapters are made up of a collection of scraps, which can be plaintext, LaTeX, or images.

TECHNOLOGIES USED

- Aurelia JavasScript Framework that provides a single page application through the use of a page router.
- Git backend for managing each scrap, chapter, and textbook
- NodeJS server interfacing between backend and user application to manage authentication and user relations

When a textbook has been created or remixed, the user is able to create a PDF of their textbook to distribute to their students or other intended audience.



A brief history of development (bottom to top): Low Level Prototype, Medium Level Prototype, and Alpha Implementation

WHAT IS REMIXING?

The idea of "remixing" is that it is not always necessary to rewrite an entire document to cover what is needed for a course.

Instead, the MVP Platform encourages a more publicly available publishing concept where documents aren't necessarily a single person's writing, but "open source" documents edited by whomever wants to change them to fit their needs.

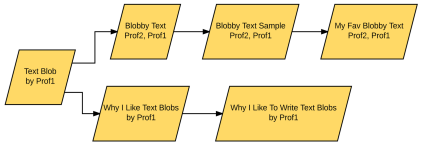


Figure: the “tree” of a paragraph being modified by multiple professors

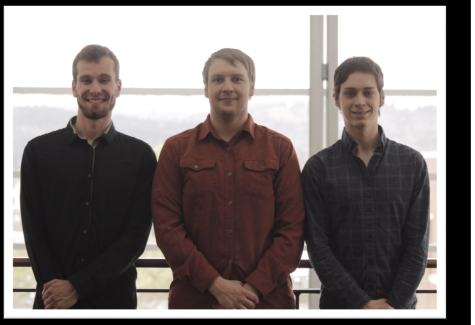
HOW DOES IT WORK?

The individual parts making up the books and their chapters are all stored separately.

Each individual paragraph is then individually modifiable - so if a professor wants to change a particular paragraph, it can be stored separately and then referenced in a new copy of that chapter.

This is done using a technology called “git”, which allows us to store a tree of changes. This means that we can keep old versions of the paragraph around for those who want them, and allow new users to modify the paragraph however they want.

CS Capstone Team #61
Team Remix



Team Members

Josh Matteson
mattesoj@oregonstate.edu

Steven Powers
powersst@oregonstate.edu

Evan Tschuy
tschuye@oregonstate.edu

Project Advisors
D. Kevin McGrath
Dr. Kirsten Winters
Jon Dodge

Project Sponsor
Dr. Carlos Jensen
jensenca@oregonstate.edu

The Many Voices Publishing Platform allows professors from all walks of life to have an equal opportunity to create the textbook they desire using textbook materials from peers around the world or create them from scratch.

Oregon State UNIVERSITY

1 PROJECT DOCUMENTATION

1.1 How does it work?

1.1.1 Project Structure

The overall structure of our project relies heavily on JavaScript. Our backend is built on a NodeJS, which handles data to and from a MongoDB database. Our frontend is comprised of Aurelia, which is a JavaScript framework used for single page applications. All three of these components make up the structure of our application, and are constantly being used by each other.

1.1.2 Theory of Operation

The MVP Platform is an online application, so navigating to the website is the first step. A user can then sign in using their Google Account, at which point they'll be able to use our application. The whole of our application works in a simple manner, as all a user has to do to get started is click on the "new book" button on the main page. Users can also fork a book on the website if they wish to work off somebody elses work. The next step is to start creating chapters, and then individual scraps from those. A user can add images, paragraphs, or other work to a scrap. The option to take others' scraps and chapters is available as well. When the book is finished, having it reviewed and printed are the only parts left to accomplish.

1.2 Hardware & Software Requirements

1.2.1 Hardware Prerequisites

The MVP Platform is a web application with search and user functionalities requiring a database. As such, it requires a decently powerful machine.
The platform has been successfully used on a machine with

1.2.2 Software Prerequisites

To begin setup of the MVP Platform, there are a few things that need to be installed.

- NodeJS and NPM: NodeJS is the runtime for the application itself, and NPM is the package manager included with it for installing 3rd-party libraries.
- MongoDB: This is the database used to store things like user favorites and login tokens.
- ElasticSearch: this is the search engine behind the search feature.

1.3 Installation Instructions

1.3.1 Setup

1) Clone the source code:

```
$ git clone https://github.com/CS461-MVP/mvp_platform.git
```

2) Move to the source code directory:

```
$ cd mvp_platform/dev
```

3) Install all scrap library dependencies:

```
$ cd scrapjs && npm install && cd ..
```

4) Install all backend Node dependencies:

```
$ cd backend && npm install
```

1.3.2 Running the backend

To run the server, simply run:

```
$ npm start
```

This will start the server running on localhost, port 8000. To verify that the server is running, in another terminal window run:

```
$ curl http://localhost:8000
```

This will return a 404, as there is no root page, if the server is successfully running.

1.3.3 Running the frontend

The frontend is a static, single-page webapp. To serve this, a standard `nginx` or `Apache` static file server is recommended; alternatively, for testing, the pages can be served with Python's built-in HTTP server:

```
$ cd /path/to/clone/mvp_platform/dev/frontend
$ python -m SimpleHTTPServer 8080
```

The website may now be visited at `http://localhost:8080`.

1.3.4 Deploying for Production

For detailed instructions regarding setting up `Nginx`, creating a service, and setting up a reverse proxy, please find the nearest sysadmin. Such a setup is outside of the scope of this documentation. In brief:

1) Create a service that runs the backend as shown above.
2) Install `nginx`.
3) Create a configuration file that:
    - Create a rule that matches requests sent to `/accounts`, `/login`, `/favorites`, `/books`, `/chapters`, `/scraps`, `/images`, `/search`, `/users`. In this rule, reverse proxy the request to the instance of the platform running on port 8000.
    - For all other requests, serve static files from `/dev/frontend`.
4) Restart `nginx` to put the rule into effect.

2 HOW DID WE LEARN NEW TECH?

By far the leading factor of our team learning new technology was having each other as knowledge points. Some members of the team had previous knowledge working with a JavaScript framework, and others had experience

working on a backend. One of the most beneficial qualities of our team was our constant communication. This made learning and asking questions very easy and fast, and aided in our own growth as developers.

Another major learning point was following documentation online. Unfortunately, the Aurelia framework isn't as polished as other JavaScript frameworks that have been around longer. This posed as a minor problem, but didn't stop us from accomplishing everything the team needed to.

1 STRUCTURES

There are several standardized structures used by the API.

1.1 `book`

Books have the following fields:

- `name`: the name of the book
- `author`: the author of the book
- `uuid`: the id of the book (immutable)
- `chapters`: a list `chapter_instance`s.

1.1.1 `chapter_instance`

A chapter instance is a specific chapter at a specific commit. This is used by the `Book` object to reference the exact version. It is a list containing:
[`chapter_uuid`, `author`, `commit sha`, `chapter_name`]
The first two fields and the last one are required. The commit sha can be `null` if the chapter is not pinned to a specific version.

1.2 `chapter`

Chapters have the following fields:

- `name`: the name of the chapter
- `author`: the author of the chapter
- `uuid`: the id of the chapter (immutable)
- `scrap`: a list of `scrap_instance` structures.

1.2.1 `scrap_instance`

A scrap instance is a specific scrap at a specific commit. This is used by the `Chapter` object to reference the exact version. It is a list containing:
[`scrap_uuid`, `author`, `commit sha`]
The first two fields are required. The commit sha can be `null` if the scrap is not pinned to a specific version.

1.3 `scrap`

Scraps have the following fields:

- `author`: the author of the book
- `uuid`: the id of the book (immutable)
- `text`: the text of the scrap

2 MAKING POST REQUESTS

Any request that writes data to the server must be authenticated. Authentication is handled with an `Authorization: Token {token}` header.

After login, the `onSignIn` function will be given a Google oAuth ID token. `POST` this ID token to the server to receive a token for this service. If in doubt, see the example page code for details.

3 BACKEND API CALLS

3.1 /books

3.1.1 *GET /books/{author}*

Returns all books for a user.

```
[
  {
    "name": "Plumbus",
    "author": "hagrid",
    "uuid": "19b66178-856d-4dad-bbc2-a9575ecfd36b",
    "chapters": [
      [
        "hagrid",
        "aec55377-716d-4274-b006-44913f73ca7f",
        null,
        "Chapter Name"
      ],
      ...
    ]
  },
  ...
]
```

3.1.2 *GET /books/{author}/{uuid}*

Returns the information about a single book, given by author and uuid.

```
{
  "name": "Plumbus",
  "author": "hagrid",
  "uuid": "19b66178-856d-4dad-bbc2-a9575ecfd36b",
  "chapters": [
    [
      "hagrid",
```

```
      "aec55377-716d-4274-b006-44913f73ca7f",
      null,
      "Chapter Name"
    ],
    ...
  ]
}
```

3.1.3 POST /books/{author}/{uuid}

Updates a book with information from the request body. Fields that can be updated are:

- `chapters`
- `name`

Only fields being updated need to be included. If updating chapters, existing chapters WILL be overwritten; include existing chapters that you do not wish to remove.
The chapters must be a list of lists, with the inner lists containing:

1) author
2) chapter uuid
3) commit sha

3.1.3.1 Example: change name and change chapter uuid / pin chapter:

```
POST /books/hagrid/19b66178-856d-4dad-bbc2-a9575ecfd36b
{
  "name": "New Book Name",
  "chapters": [
    [
      "hagrid",
      "68c47c74-f6fb-4e5b-a68c-f2c6b4265bd1",
      "ad3df920f3ce04687732c5572a76e541e6a1b799"
    ]
  ]
}
```

`POST` update requests return the message body of the new commit:

```
{
  "message": "update:..."
}
```

3.1.4 *POST /books/{author}/{uuid}/fork*

TODO

3.1.5 *GET /books/{author}/{uuid}/pdf*

Returns a PDF file containing the book.

3.1.6 *GET /books/{author}/{uuid}/history*

Returns the history of the book.

```
[
  [
    "a632575d435f86e19434092fe3e2fca98a3bb7e5",
    "update: changed name from Hagrid's Big Book of BBQ to New Book Name. "
  ],
  [
    "b76c2469b00ee63c0e462faa32a5875abbbd51c7",
    "update: updated chapters (TODO diff). "
  ],
  [
    "15ccb3513d209ca9770cb188ace7fe9bcd97b433",
    "Created book named Hagrid's Big Book of BBQ"
  ]
]
```

3.1.7 *POST /books/new*

Create a new book object. The name and author are both required; chapters cannot be added. The author must be the same as the logged in user.

```
POST /books/new
{
  "name": "My Favorite Book",
  "author": "magellan"
}
```

`POST` creation requests return a book structure:

```
{
  "name": "My Favorite Book",
  "author": "magellan",
  "uuid": "802619b6-f102-4b63-9fbe-f578c7af671f",
  "chapters": []
}
```

3.2 `/chapters`

3.2.1 *GET /chapters/{author}*

Returns all chapters for a user.

```
[
  {
    "author": "hagrid",
    "chapter one",
    "uuid": "68c47c74-f6fb-4e5b-a68c-f2c6b4265bd1",
    "scraps": [
      [
        "hagrid",
        "f47c9536-7e72-4fd1-9532-282cedfddc72"
      ],
      ...
    ]
  },
  ...
]
```

3.2.2 *GET /chapters/{author}/{uuid}*

Returns information about a single chapter, given by author and uuid.

```
{
  "author": "hagrid",
  "name": "chapter one",
  "uuid": "68c47c74-f6fb-4e5b-a68c-f2c6b4265bd1",
  "scraps": [
    [
      "hagrid",
      "f47c9536-7e72-4fd1-9532-282cedfddc72"
    ],
    ...
  ]
}
```

3.2.3 *POST /chapters/{author}/{uuid}*

Updates a chapter with information from the request body. Fields that can be updated are:

- `scraps` (a list of `scrap_instance`s)

- `name`

Only fields being updated need to be included. If updating scraps, existing scraps WILL be overwritten; include existing scraps that you do not wish to remove.

3.2.3.1 Example: change name and replace scrap / pin scrap:

```
POST /scraps/hagrid/19b66178-856d-4dad-bbc2-a9575ecfd36b
{
  "name": "New Chapter Name",
  "scraps": [
    [
      "hagrid",
      "68c47c74-f6fb-4e5b-a68c-f2c6b4265bd1",
      "ad3df920f3ce04687732c5572a76e541e6a1b799"
    ]
  ]
}
```

`POST` update requests return the message body of the new commit:

```
{
  "message": "update: ..."
}
```

3.2.4 POST /chapters/{author}/{uuid}/fork

TODO

3.2.5 GET /chapters/{author}/{uuid}/pdf

Returns a PDF file containing the chapter

3.2.6 GET /chapters/{author}/{uuid}/history

Returns the history of the chapter.

```
[
  [
    "3c27eaa113f876ba389f27835478f0b09c60aa39",
    "update: changed name from Second Part to Second Stanza. "
  ],
  [
    "018a57d8b72a131873ee0534c42c47bf8abc5c25",
    "update: updated scraps (TODO diff). "
  ],
  [
```

```
    "61de182a1b638611633f829e98f35e6423038be5",
    "Created chapter named Second Part"
  ]
]
```

3.2.7 *POST /chapters/new*

Create a new chapter object. The name and author are both required; scraps cannot be added. The author must be the same as the logged in user.

```
POST /chapters/new
{
  "name": "My Favorite Chapter",
  "author": "magellan"
}
```

`POST` creation requests return a chapter structure:

```
{
  "name": "My Favorite Chapter",
  "author": "magellan",
  "uuid": "802619b6-f102-4b63-9fbe-f578c7af671f",
  "scraps": []
}
```

3.3 `/scraps`

3.3.1 *GET /scraps/{author}*

Returns all scraps for a user.

```
[
  {
    "author": "hagrid",
    "text": "But then someone shouted...\"Hey, that's Argentina!\"\nMagellan...",
    "uuid": "1a530c40-d8ed-4362-b402-4481c50e50a2"
  },
  ...
]
```

3.3.2 *POST /scraps/{author}/{uuid}*

Updates a scrap with information from the request body. Fields that can be updated are:

- `text`
- `latex`
- `tags`

3.3.2.1 Example: change text, latex, tags:

```
POST /scraps/hagrid/1a530c40-d8ed-4362-b402-4481c50e50a2
{
  "text": "hello world!",
  "latex": false,
  "tags": [
    "tag1",
    "tag2",
    "tag3"
  ],
}
```

`POST` update requests return the message body of the new commit:

```
{
  "message": "update: changed text from This is a text to hello world!."
}
```

3.3.3 POST /scraps/{author}/{uuid}/fork

TODO

3.3.4 GET /scraps/{author}/{uuid}/pdf

Returns a PDF file containing the scrap

3.3.5 GET /scraps/{author}/{uuid}/history

Returns the history of the scrap.

```
[
  [
    "aa62f9f63c013795ee69c2dec3321593b8fbbecd",
    "update: changed text from Lovely sentence about flowers to Hello World."
  ],
  [
    "0e0b4ef415bc3cbd6e4ed004d719b611c8fc657c",
    "update: changed text from to Lovely sentence about flowers."
  ],
  [
    "2fc3a0061705f34c6d559b75085c895751b815a3",
    "Created new scrap"
  ]
]
```

3.3.6 *POST /scraps/new*

Create a new scrap object. The author is required; text can be added. The author must be the same as the logged in user.

```
POST /scraps/new
{
  "author": "magellan"
}
```

`POST` creation requests return a scrap structure:

```
{
  "author": "magellan",
  "uuid": "802619b6-f102-4b63-9fbe-f578c7af671f",
  "text": ""
}
```

3.3.7 *POST /images/new*

Create a new image. This is a `form/multipart` file upload; the image must be in the field "image". When successful, it will return a scrap object that can be included in a chapter to include the picture:

```
{
author: "hagrid",
text: "\includegraphics[width=\textwidth]{/images/hagrid/mhcf7dezlahdxu5jt6gvi}",
latex: true,
image: true,
isNew: true,
uuid: "266bfe4c-2338-4a03-afdd-a9765fdbe629"
}
```

3.4 `/search`

Search:

```
/search?q=url%20encoded%20query
```

Search for specific types:

```
/search?q=abc123&type=books
/search?q=abc123&type=chapters
/search?q=abc123&type=books&type=chapters
```

Search only hagrid's items (can specify any user):

```
/search?q=abc&user=hagrid
```

3.5 `Favorites`

There are three API calls in the favoriting family.

3.5.1 *POST /{type}/{author}/{id}/favorite*

Adds a favorite. If the object is already favorited, it stays in the favorites. The return body is empty; the return status code is 204.

Ex: `POST /books/hagrid/19b66178-856d-4dad-bbc2-a9575ecfd36b/favorite`

3.5.2 *DELETE /{type}/{author}/{id}/favorite*

Removes a favorite. If the object isn't favorited, it stays not favorited without an error. The return body is empty; the return status code is 204.

Ex: `DELETE /books/hagrid/19b66178-856d-4dad-bbc2-a9575ecfd36b/favorite`

3.5.3 *GET /favorites*

Gets all of the user's favorites. The token is used to determine whose favorites to return.

```
[
  {
    "type": "book",
    "author": "evantschuy",
    "uuid": "19b66178-856d-4dad-bbc2-a9575ecfd36b"
  },
  {
    "type": "book",
    "author": "hagrid",
    "uuid": "19b66178-856d-4dad-bbc2-a9575ecfd36b",
    "name": "Plumbus: How They're Made"
  },
  {
    "type": "scrap",
    "author": "hagrid",
    "uuid": "31f54157-1ab4-48fc-a606-849e28270faf",
    "text": "this is a new scrap created 14:03 13 April 2017"
  }
]
```

To filter to specific types, use the same filter as for `search`: `?type=...`:

```
GET /favorites?type=book

[
  {
    "type": "book",
    "author": "evantschuy",
    "uuid": "19b66178-856d-4dad-bbc2-a9575ecfd36b"
```

```
  },
  {
    "type": "book",
    "author": "hagrid",
    "uuid": "19b66178-856d-4dad-bbc2-a9575ecfd36b",
    "name": "Plumbus: How They're Made"
  }
]
```

Multiple types can be specified:

```
GET /favorites?type=book&type=chapter
```

3.6 /accounts/myaccount

3.6.1 POST /accounts/myaccount

Provides an endpoint to update the name to be displayed on book renders:

```
{"name": "D. Campbell"}
```

3.6.2 GET /accounts/myaccount

Provides an endpoint to view the name to be displayed on book renders:

```
{
  "userid": "dcampbell",
  "name": "D. Campbell"
}
```

Many Voices Publishing Platform User Guide

June 2017

Version 1.10

Team Remix
Josh Matteson
Steven Powers
Evan Tschuy

Table of Contents

1 Introduction

1.1 Scope and Purpose

*The Many Voices Publishing Platform (MVP Platform) is a document creation tool aimed at professors looking to write a textbook for their courses but are short on availability. The MVP Platform allows a user to create their own **Textbooks**, **Chapters**, or **Scraps** or copy and modify existing textbooks, chapters, or scraps created by other users with the MVP Platform.*

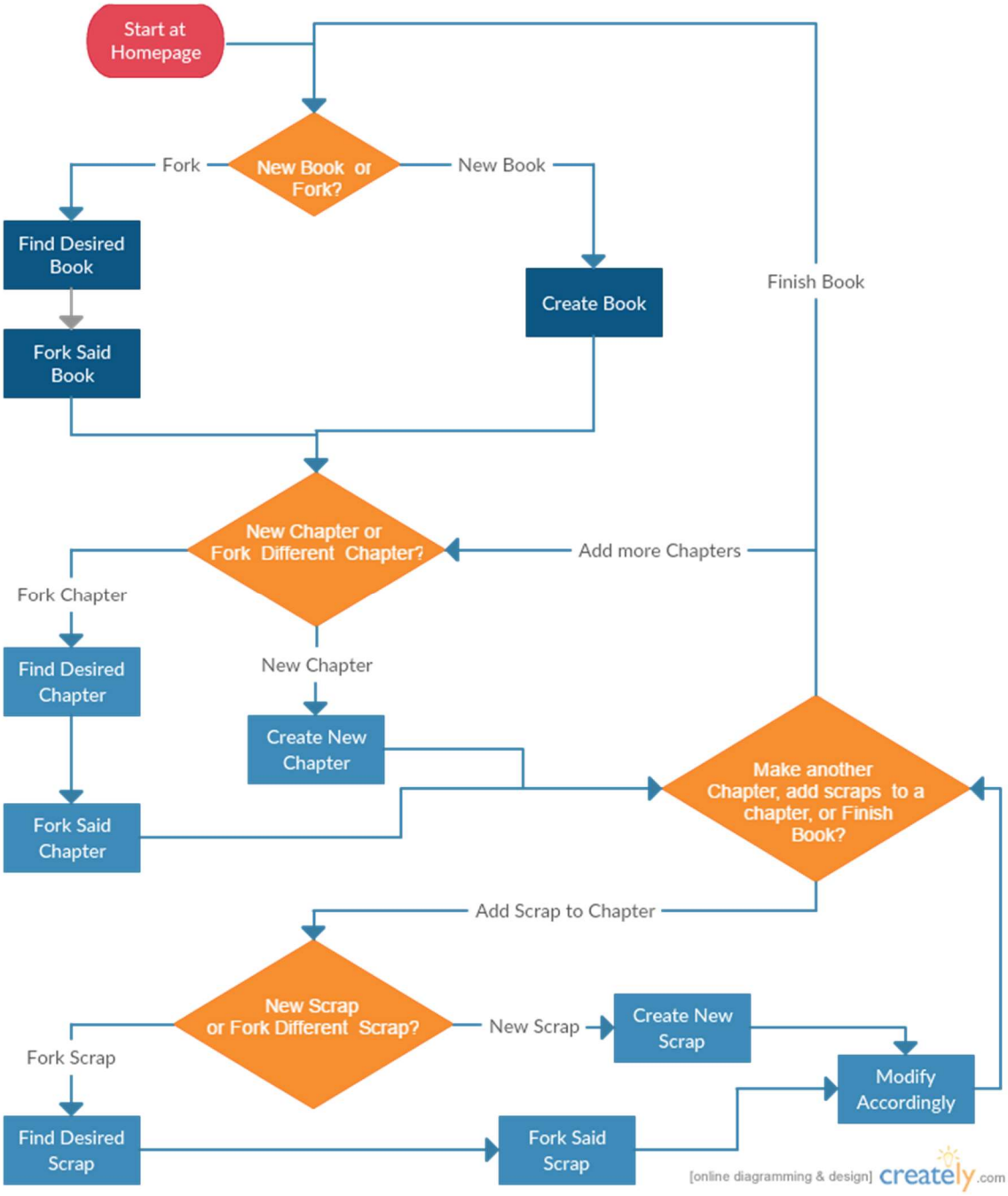
This user guide will provide basic guidelines for account login, textbook creation, editing a textbook, chapter creation, editing a chapter, scrap creation, creating an image scrap and profile settings.

1.2 Process Overview

A user will typically follow the order of operations listed below, this user guide will cover these operations in this order.

1. New Textbook
2. Edit Textbook
3. New Chapter
4. Edit Chapter
5. New Scrap
6. New Image Scrap
7. Profile Settings

Process Flow Diagram:



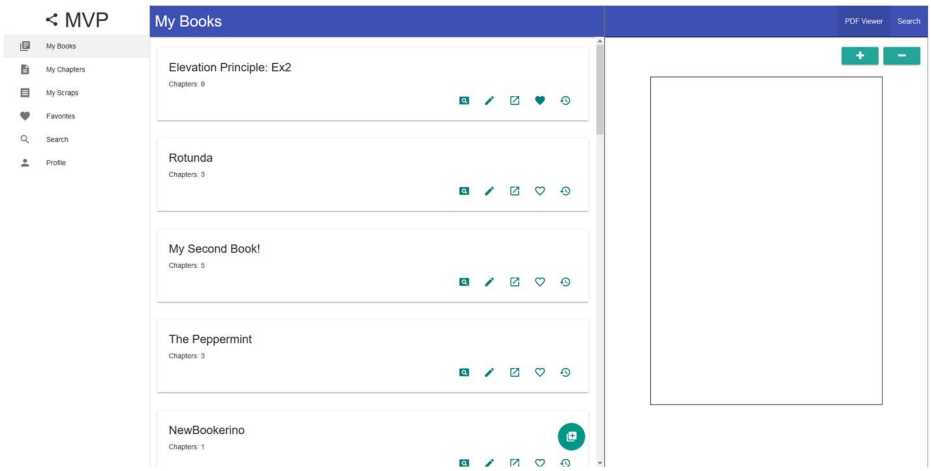
2 New Textbook

Create a new textbook for your text creation or modification to take place.

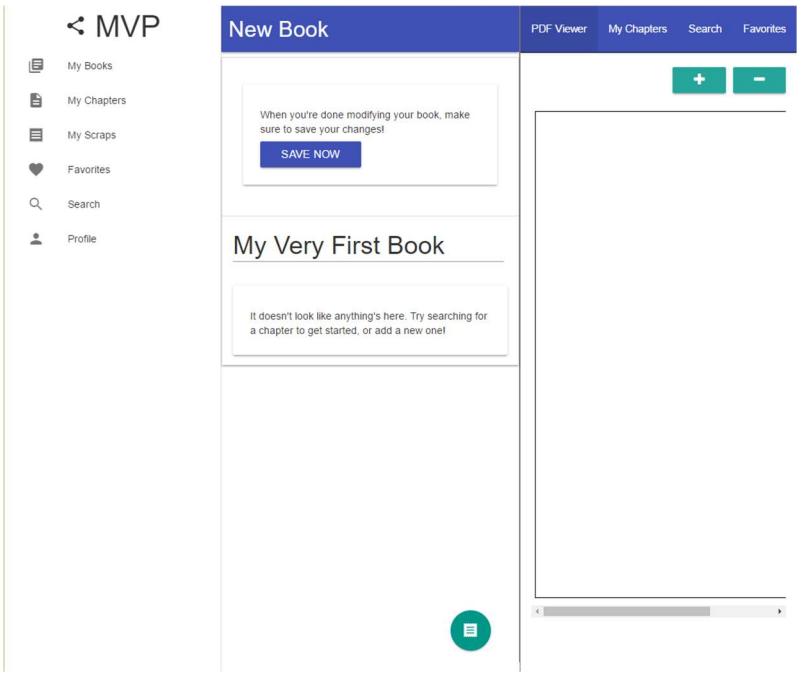
The new textbook operation is a simple press of a button which will prompt you for a new book name.

2.1 New Textbook

1. From the My Books page locate the Create New Book button



2. Click the Create New Book button
3. Input a book name



4. Click “Save Now”

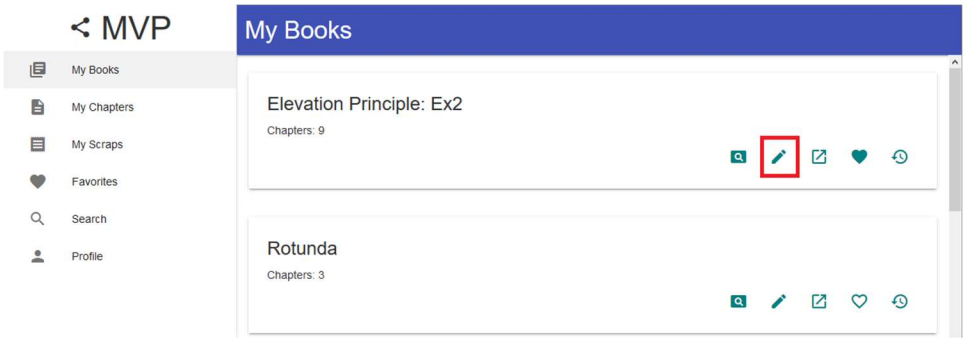
3 Edit Textbook

Edit an existing textbook for your text creation or modification to take place.

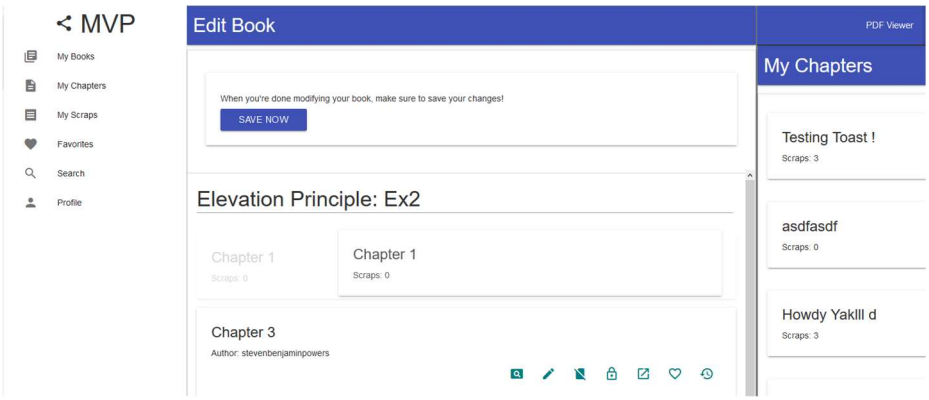
The edit textbook operation is a simple press of a button which will allow the user to edit the title of their book, rearrange chapters, search and drag chapters into their book.

3.1 Edit Textbook

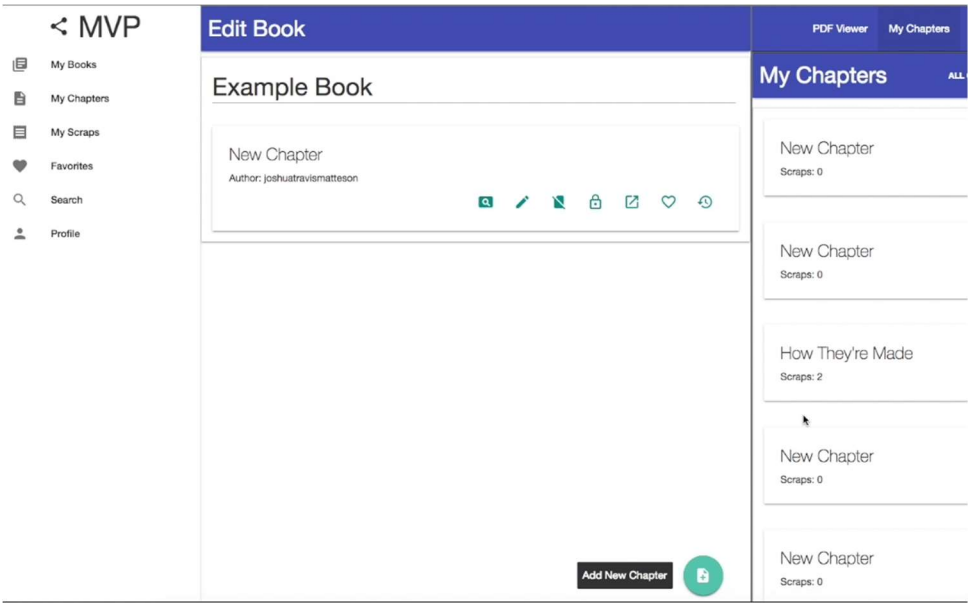
1. From the My Books page locate the Edit Book button



2. Click the Edit Book button
3. Options:
    a. Drag in a new chapter from My Chapters / Searched chapters



    b. Rename the book
        i. Click on the book title to rename the book
    c. Create a new chapter



d. You will be prompted to enter a name for the chapter
e. For more details see section 4

4. Click “Save Now”

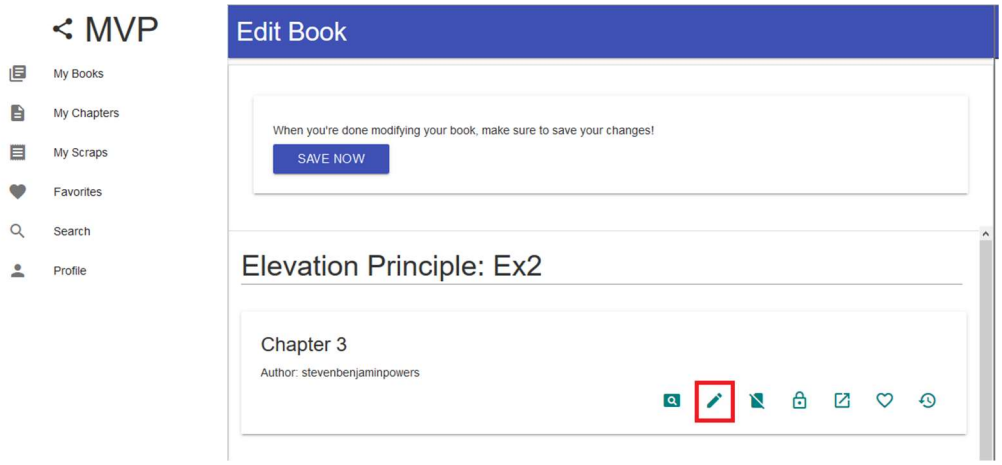
4 Edit Chapter

Edit an existing chapter for your text creation or modification to take place.

The edit chapter operation is a simple press of a button which will allow the user to edit the title of their chapter, rearrange scraps, search and drag scraps into their book.

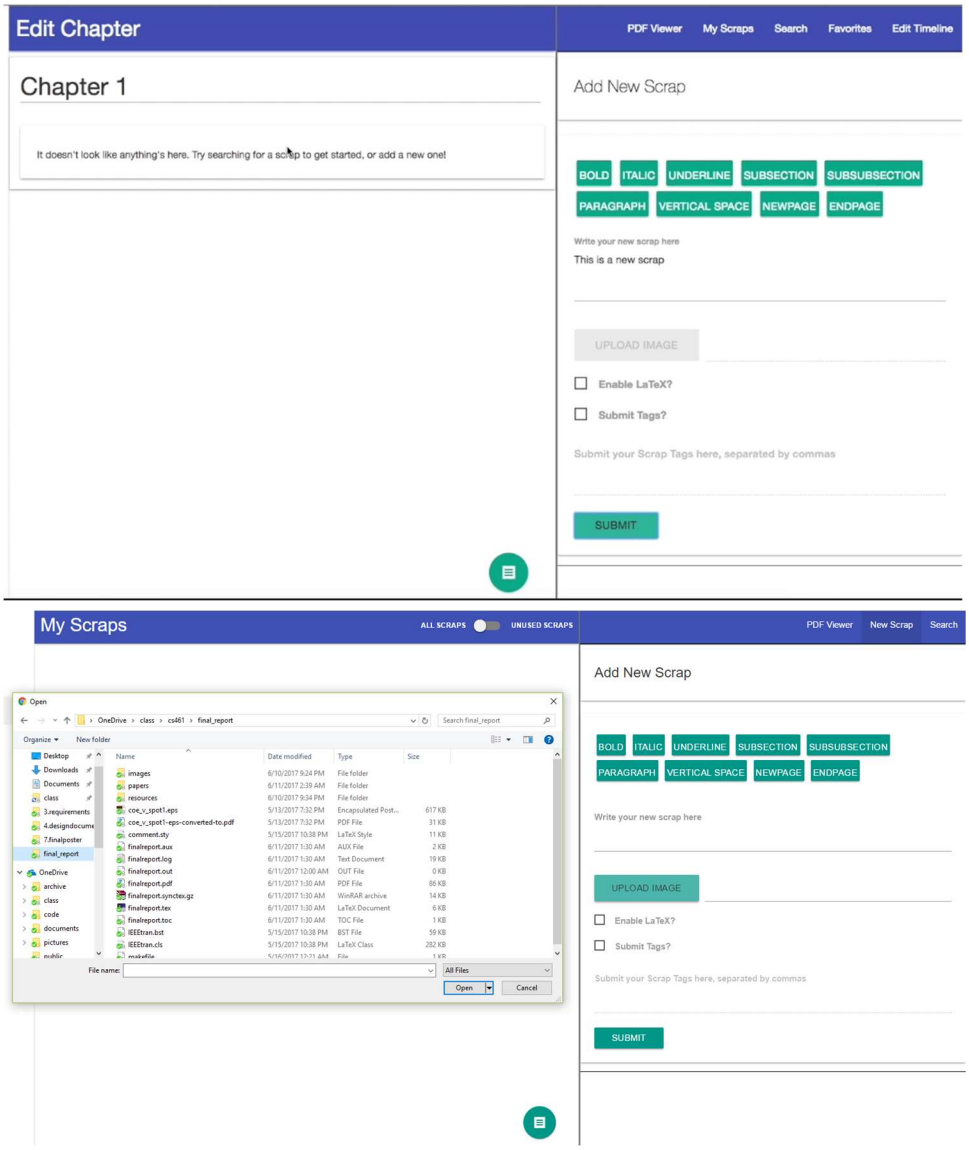
4.1 Edit Chapter

1. From the Edit Book page locate the Edit Chapter button



2. Click the Edit Chapter button
3. Options:
    a. Drag in a new scrap from My Scraps / Searched scraps

    b. Rename the chapter
        i. Click on the chapter title to rename the chapter
    c. Create a new scrap



d. You will be prompted to enter either text or upload an image for the scrap
e. For more details see section 5

4. Click “Save Now”

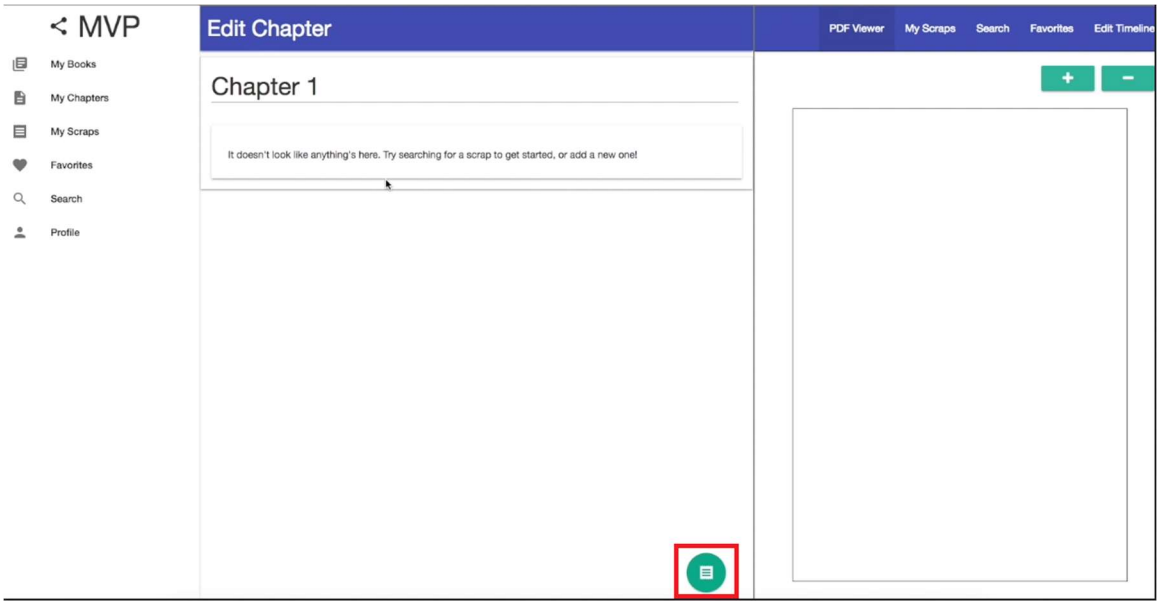
5 New Scrap

Create a new scrap that chapter for your text creation or modification to take place.

The edit chapter operation is a simple press of a button which will allow the user to edit the title of their chapter, rearrange scraps, search and drag scraps into their book.

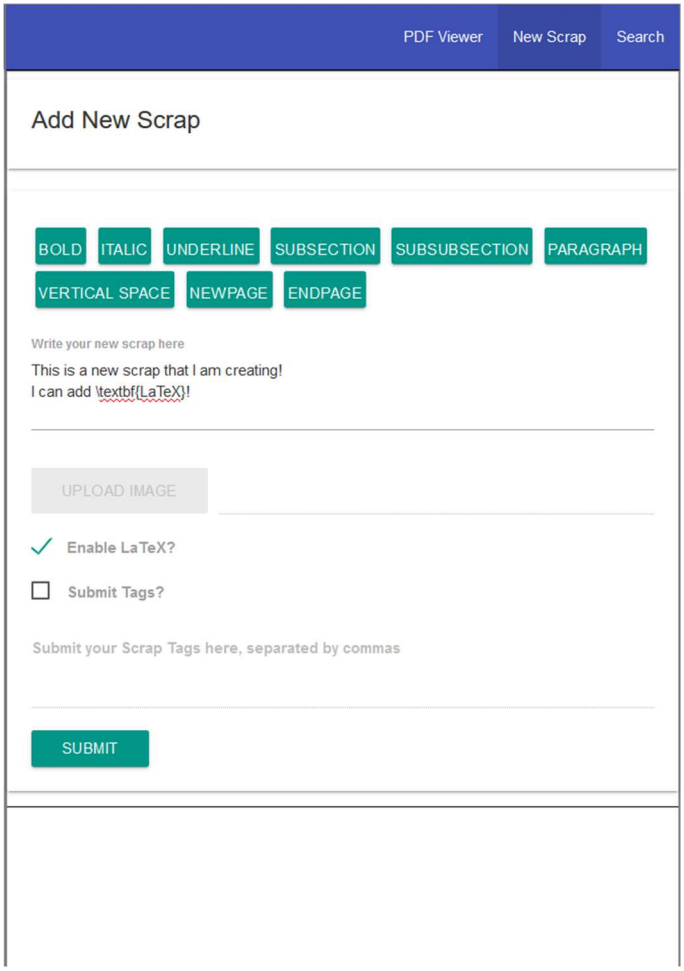
5.1 New Scrap

1. From the Edit Chapter page locate the New Scrap button

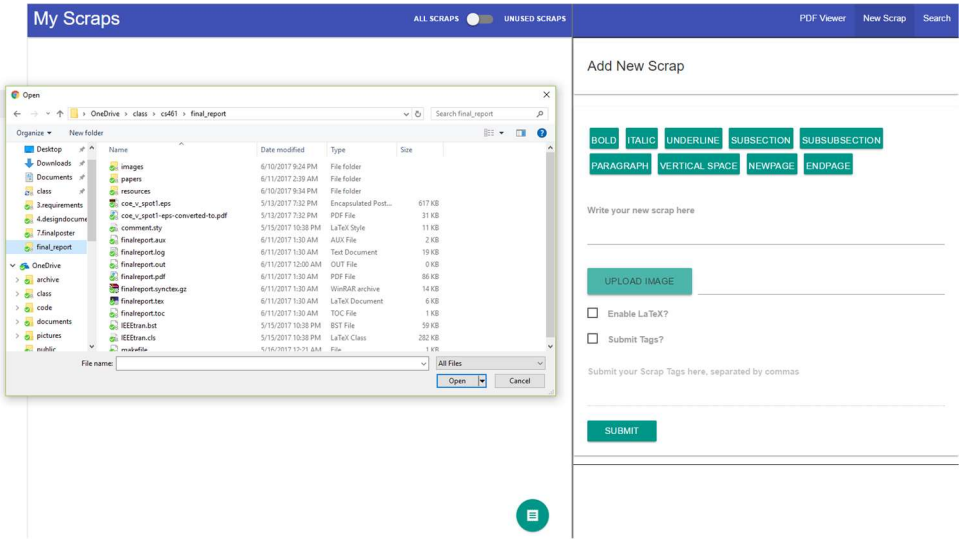


2. Click the New Scrap button

3. Options:
    a. Create a new scrap using text, enable LaTeX mode, or add tags



    b. Or create a new image scrap



4. Click “Save Now”

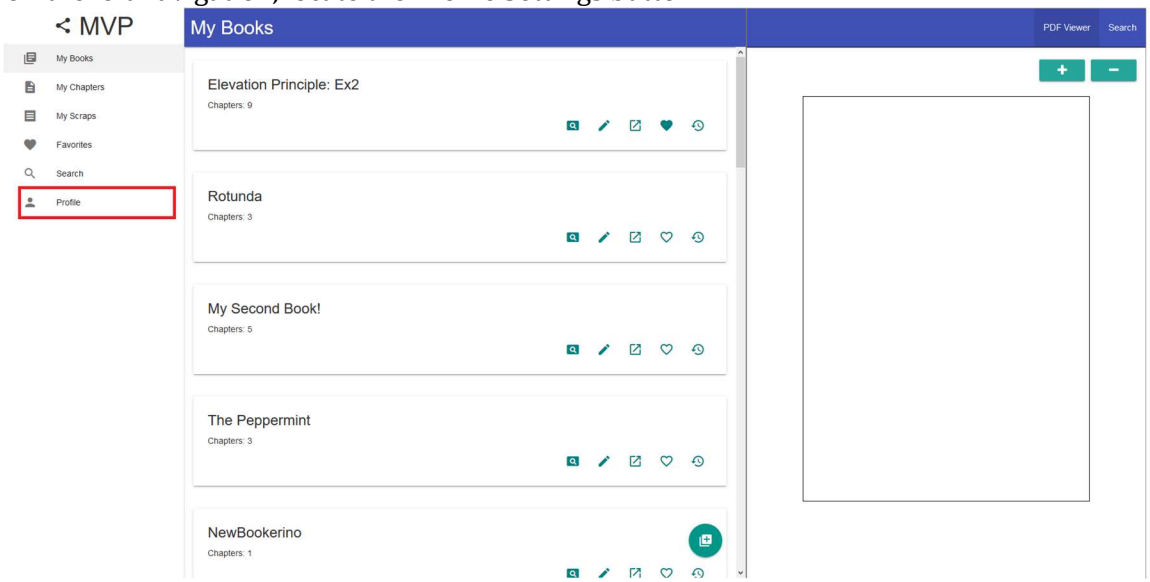
6 Profile Settings

Profile settings is a way to alter the name that appears on compiled documents or logout.

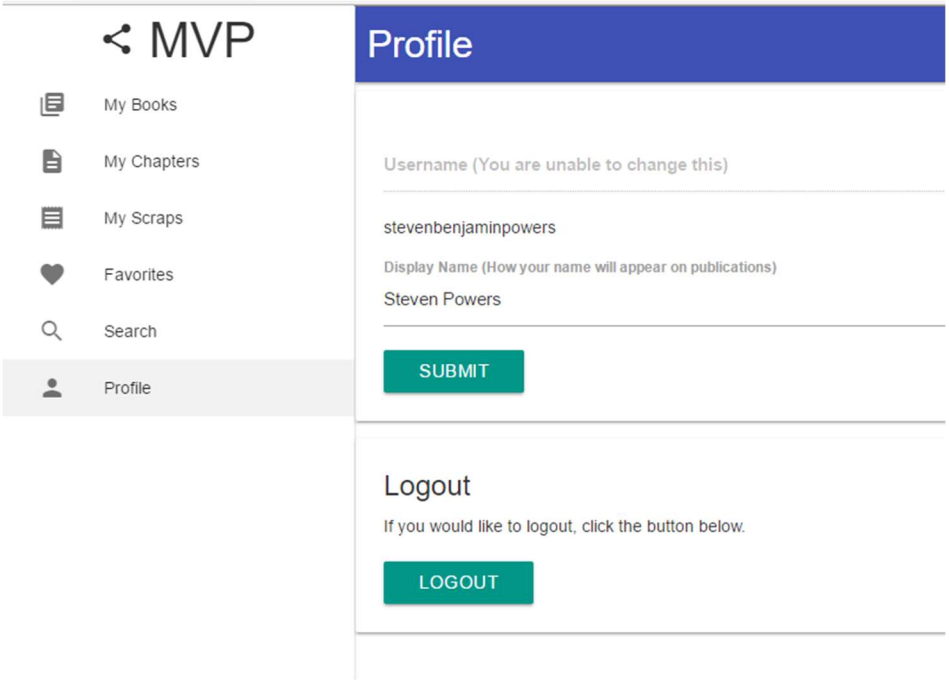
The profile settings page allows the user to view their account information, modify the name that appears on their compiled documents, or logout for another

6.1 Profile Settings

1. On the left navigation, locate the Profile Settings button.



2. Click the Profile Settings button
3. Options
    a. Modify your display name and click “Submit” or logout by clicking “Logout”



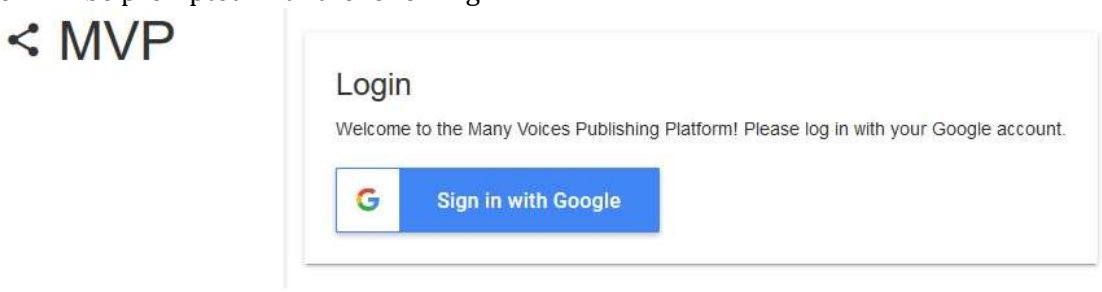
7 Login

Login functionality is an essential part of any web application .

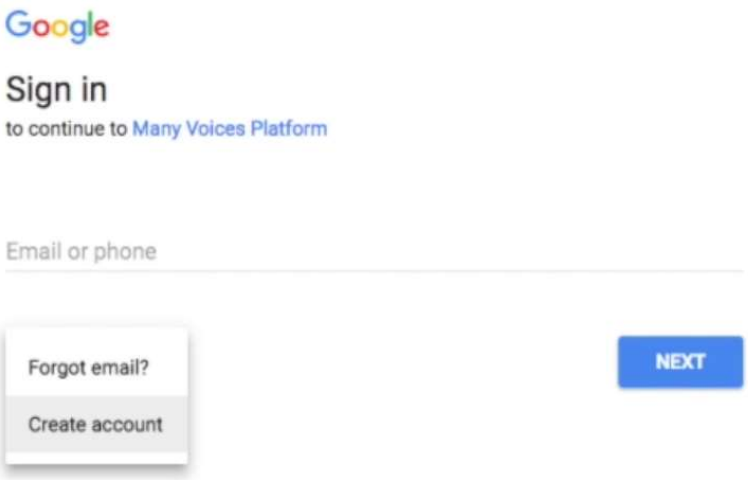
The profile settings page allows the user to view their account information, modify the name that appears on their compiled documents, or logout for another

7.1 Profile Settings

1. When a user navigates to the website for the first time, or when no login is saved the user will be prompted with the following.



2. Click the “Sign in with Google” button
3. You will be prompted to select the account you want to use or you can create a new account with Google.



   a.
4. Once logged in, you will be presented with the MyBooks view.

If you encounter issues not addressed by this user guide, please contact your account manager for additional support.

1 How did we learn new Technologies?

By far the leading factor of our team learning new technology was having each other as knowledge points. Some members of the team had previous knowledge working with a JavaScript framework, and others had experience working on a backend. One of the most beneficial qualities of our team was our constant communication. This made learning and asking questions very easy and fast, and aided in our own growth as developers.

Another major learning point was following documentation online. Unfortunately, the Aurelia framework isn't as polished as other JavaScript frameworks that have been around longer. This posed as a minor problem, but didn't stop us from accomplishing everything the team needed to.

2 What Websites Were Helpful?

Aurelia Documentation - http://aurelia.io/hub.html
Aurelia Materialize - https://aurelia-ui-toolkits.github.io/demo-materialize/
Aurelia Chatroom - https://gitter.im/aurelia/Discuss
Aurelia Materialize Chatroom - https://gitter.im/aurelia-ui-toolkits/aurelia-materialize-bridge

3 What References Books Really Helped?

There were no resource books that were utilized during this project.

4 Were There Any People On Campus That Were Really Helpful?

Beyond the professors and those related to this course of this course, there were no really helpful people on campus.

5 WHAT I LEARNED: JOSH MATTESON

I believe the biggest thing I learned throughout the year is how crucial it is to have communication between everybody involved, whether that be other team members or our client. In the beginning of the year, there wasn't a whole of communication on anyones part. This caused for a lot of frustration, but thankfully it got better as we continued on.

5.1 What technical information did I learn?

While I already had a decent understanding of web applications are made, I feel that this skill was strongly reinforced by working on this project. JavaScript is one of the major languages that companies are looking for, and our entire application was built around JavaScript.

Another technical skill that I learned over the year is how to write papers in LaTeX. Our application is centered around books being built, and they are all stored in the backend as LaTeX. This skill proved greatly useful for other classes that required a paper with illustrations or math problems.

Many factors go into building a web application, and it's hard to nail down every I learned from it. Web development is one of the more popular desired fields of computer science, and having knowledge in this area greatly influences my chances of getting a job after college.

5.2 What non-technical information did I learn?

I would say the largest piece of non-technical information I've learned is how to work with a team and how to communicate properly. Organizing team meetings and following through with it is something I've learned as an important factor in the success of any group activity. Another important non-technical skill I've learned is learning to set deadlines and having other team members keep you accountable. Making sure that everyone is getting their work done on time is one of the biggest factors of success, as long as it doesn't come in the form of micromanaging.

5.3 What have I learned about project work?

This is a repeated theme throughout my writing, but communication is key. From the beginning, a member of the team set up a communication platform that we could all communicate over. I think this is largely why our team was able to do so well and actually have a working demo at expo. I've also learned that sometimes you just have to do more than you thought you would in order to succeed. This comes in the form of picking up the slack of other people, and sometimes other people pick up your slack.

5.4 What have I learned about project management?

Organizing everything beforehand and setting up deadlines is crucial to good project management. Otherwise, the project will get away from you and you'll be left wondering what happened.

5.5 What have I learned about working in teams?

I've learned that working in teams is a lot of give and take. Sometimes you have to rely on your teammates for certain task, and other times your teammates rely on you to get everything done. This is imparative to having a functioning team, because if you can't rely on your team to work, then there's no trust.

5.6 If you could do it all over, what would you do differently?

If I could redo the project differently, I would get a lead on the project as soon as possible. We started the biggest parts of the projects later than we should have, and this costed us when time was crunched for other classes and projects. I know that, for me personally, I could have been researching far more than I had early on. This could have saved us an incredible amount of time when we were stuck on various technical problems about the Aurelia framework.

6 WHAT I LEARNED: STEVEN POWERS

6.1 What technical information did I learn?

I learned a lot of technical information during the past year. As a result of the year of working on this project I have learned a lot more about newer technologies such as page routers. I also learned a lot about NodeJS, containers, and how JavaScript frameworks such as Aurelia are different than raw JavaScript that I am used to working with. Aurelia Materialize and Aurelia Single Page Applications are two examples of some of the front end technologies that I learned during development.

6.2 What non-technical information did I learn?

I learned a lot about the documentation portion of software development projects. The various document formats, specific IEEE standards specifications, etc. I also learned about how to write these documents and write about my project at a higher level. I also learned how to present myself and talk about projects I am working on with someone without a technical background.

6.3 What have I learned about project work?

I have learned a lot about how projects can become disorganized or put in disarray when some information is unknown. No matter how much planning is put into the various documents detailing your targeted goals and dates of delivery, this can cause your workflow to drastically change.

6.4 What have I learned about project management?

Managing a project is a tricky one. I was elected / volunteered to be team leader and as a result I had to stay extra vigilant on deadlines and make sure that things were getting finished on time.

6.5 What have I learned about working in teams?

Everyone has different strengths and weaknesses, and being understanding of the weaknesses and exploiting their strengths can still result in a successful project.

6.6 If you could do it all over, what would you do differently?

If I could start over and do it all over again, I probably wouldn't. There was a lot of time and effort put into this course and I wouldn't want to suffer with that right now. If I could change something that would put me into a better situation now, without effort, it would be to start working on the project earlier. We were put into a bad situation and had to wait before we could continue with development early on in Winter term. While other groups continued plugging away at their projects, we were trying to get clarification and guidance from our client. Working with improved client relations, such as how it was later in the term, and spring term. For the project itself, I would have tried using a different JavaScript framework, or see how much we could have got completed with just the addition of a router instead of a whole framework that we had to learn how to use.

7 WHAT I LEARNED: EVAN TSCHUY

At times fun, at times incredibly frustrating, the last year of capstone has been a great opportunity for me to get a look at the perils of project management, and dig into creating a technical product from nothing.

7.1 What technical information did I learn?

On a technical level, the project can be broken down into several sections:

- a NodeJS backend:
  - direct interactions with ElasticSearch
  - a custom-written library for interacting with `git` repositories
  - 3rd-party authentication integration with Google
- a frontend:
  - a single-page application using Aurelia
  - Materialize for user interface items

Every single one of these sections was its own learning experience for me. In the past, I have written web applications for work. These generally tended to be fairly self-contained, with simple user interfaces that would fit in well with the year 2009.

Instead, this project was hyper-modern. We purposefully chose a single-page application backed by a modern backend, a complicated datastore, and a powerful, real-world search engine. All of these things were new to me, and being exposed to them has prepared me in the event I decide to continue creating web applications.

7.2 What non-technical information did I learn?

Some people will need pushing to get things done. There's no way around that, and so it is simply necessary to know when to push and how. It is necessary all throughout the year to make tradeoffs in who does what, when, and why.

7.3 What have I learned about project work?

It is easier if you just accept that some people put in less effort. After weighing how much I wanted to finish the project and how much recognition I would get from my group, my peers, the school, and others, I simply went ahead and did parts that we had earlier divvied up otherwise. The other option would simply have been to let the project go uncompleted, which I was not alright with.

7.4 What have I learned about project management?

I think that what I learned about project management can be summed up in one small line:

> Divide up work early, and check progress often. Readjust as needed.

7.5 What have I learned about working in teams?

Working in teams is a give and take situation. For parts of a project where you are the most technically familiar, it is often a good idea to take on that section while someone else works on things they are more familiar with. On the other hand, if you have a little more free time, it can be good to instead work together on both sections, so that you can learn what they know and they can learn what you know.

7.6 If you could do it all over, what would you do differently?

I have told every single one of my friends that are going into capstone next year the exact same thing: **make a team and contact a good project**, and **do it — don't just do the rankings on the website**. It is worth it to not get stuck with strangers on a project that you didn't choose, and you might get stuck with a project you didn't choose even if you submit your rankings. That has lead this past year to be much worse than it could've been, had I just had the energy to put in to corral some of my friends into doing a project we liked. It was an alright year but it could've been much better.

1 Code Examples

The backend is split into two primary parts. The first, and most difficult part of the backend, is the interface library being used to store objects into the git repositories and to get them back out. Shown below is a code sample, which is fully working today, showing a simple script that creates new scraps, chapters, and books, and gets a valid latex document to render from them. Below are two interesting pieces of backend code that allow the user to have a seamless experience combining different scraps, chapters, and books together.

```
var a = new scrap.Scrap('First sentence of a chapter', 'rambourg');
await a.save('initial save');

var b = new scrap.Scrap('Second sentence of a chapter', 'rambourg');
await b.save('initial save');

var ch1 = new chapter.Chapter('First Chapter Name', 'rambourg');
ch1.addScrap(a);
ch1.addScrap(b);

await ch1.save('initial save');

var c = new scrap.Scrap('First sentence of another chapter', 'rambourg');
await c.save('initial save');
var ch2 = new chapter.Chapter('Another Chapter Title', 'rambourg');
ch2.addScrap(c);
await ch2.save('initial save');

var myBook = new Book('My Fav Book Title', 'rambourg');
myBook.addChapter(ch1);
myBook.addChapter(ch2);

var bookText = await myBook.getText();
```

Another piece of interesting code that we want to highlight is the creation of a new scrap. In the code example below, we determine whether this new scrap is going to be appended to an existing chapter, and if so, we need to first return and collect all of the existing scraps associated with that chapter. Once we have the new list of scraps, we need to update the chapter object with the new information.

```
var scr = new scrap.Scrap('Brody', 'abc123-def456')

/* reconstitute a chapter from author and id */
var ch = chapter.reconstitute('Patrick Rambourg', '56c2c2ee')
var newCh = await ch.fork('Brody')

newCh.addScrap(scr)
await newCh.save('add new chapter')

submitNewScrap() {

if (this.selectedFile) {
this.submitImageScrap();
return;
}

var requested = this.userText;
var enableLatex = this.enableLatex;
var theAuthor = Cookies.get('username');
var theChapter = this.chapters[1];
var authToken = "Token " + Cookies.get('token');

var submitTags = this.submitTags;
var submittedTags = this.submitTagsText;

if (theChapter === undefined || theChapter === null || theChapter === "") {
if (submitTags) {
submittedTags = submittedTags.toString();
submittedTags = submittedTags.replace(/(^,)|(,$)/g, "");
submittedTags = submittedTags.replace(/\s/g, '');
submittedTags = submittedTags.split(",");
}
let request = {
author: theAuthor,
text: requested,
latex: enableLatex,
tags: submittedTags
};
```

The next code listing to be shown is the new scrap creation process. This is a particularly interesting section of code because it determines what to do for the user. Based on what page the user's request originated the new scrap function will either simply create the scrap and place it within the user's MyScrap page. If the user's request originated from the edit chapter page, it will dynamically build the list of all existing scraps and their ordering, and then append the new scrap and submit the new list to the database.

```
//CREATE THE NEW SCRAP AND GET THE NEW SCRAP ID
var scrapID = '';
httpClient.fetch('https://remix.ist/scraps/new', {
method: 'post',
body: JSON.stringify(request),
headers: {
'Content-Type': 'application/json',
'Authorization': authToken
}
})
.then(response => response.json())
.then(data => {
var scrapID = data.uuid;
this.toast.show('Scrap saved successfully!', 5000);
this.ea.publish('new-scrap', data);
});
} else { //The Chapter Was Provided, so create the scrap and then add it to a chapter.
if (submitTags) {
submittedTags = submittedTags.toString();
submittedTags = submittedTags.replace(/(^,)|(,$)/g, "");
submittedTags = submittedTags.replace(/\s/g, '');
submittedTags = submittedTags.split(",");
}

let request = {
author: theAuthor,
text: requested,
latex: enableLatex,
tags: submittedTags
};

//CREATE THE NEW SCRAP AND GET THE NEW SCRAP ID
var scrapID = '';
httpClient.fetch('https://remix.ist/scraps/new', {
method: 'post',
body: JSON.stringify(request),
headers: {
'Content-Type': 'application/json',
'Authorization': authToken
```

```
}
})
.then(response => response.json())
.then(data => {
var new_scrap = data;
var scrapID = data.uuid;

this.scraps = [];
var test = [];
//GET CHAPTERS SCRAPS
httpClient.fetch('https://remix.ist/chapters/' + theAuthor + '/' + theChapter)
.then(response => response.json())
.then(data => {

var scraps;
this.scraps.push(data);
scraps = data.scraps;

var newScrapRequest = [];

for (var i = 0; i < scraps.length; i++) {
newScrapRequest.push([scraps[i][0], scraps[i][1], scraps[i][2]]);
}
newScrapRequest.push([theAuthor, scrapID, null]);
let request2 = {
scraps: newScrapRequest
};

//   UPDATE CHAPTER WITH NEW SCRAPS
httpClient.fetch('https://remix.ist/chapters/' + theAuthor + '/' + theChapter, {
method: 'post',
body: JSON.stringify(request2),
headers: {
'Content-Type': 'application/json',
'Authorization': authToken
}
})
.then(response => response.json())
.then(data => {
this.toast.show('Scrap saved successfully!', 5000);
this.ea.publish('new-scrap', new_scrap);
});
}
}
```

2 PROJECT IMAGES

This section is dedicated to showing some of the progress that has been made through iteration and user feedback. Starting with our low level prototypes, medium fidelity prototypes, high fidelity prototypes, and finally our finished product. The designs were influenced by user feedback throughout the development process.

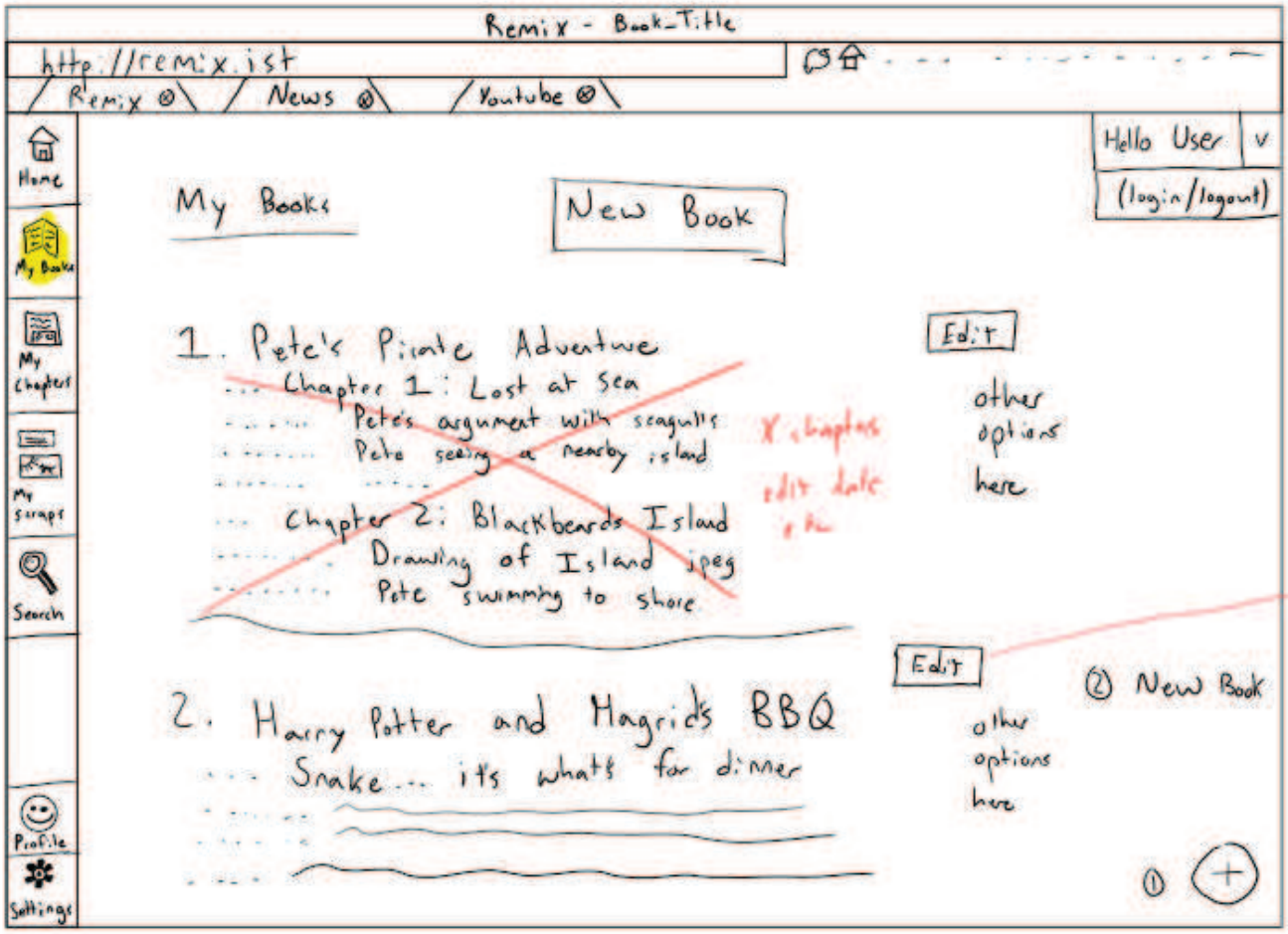


Fig. 1. Here is one low fidelity prototype that was shown to our client. Dr. Jensen suggested removing the preview of chapters and scraps to make the interface a little more simple.

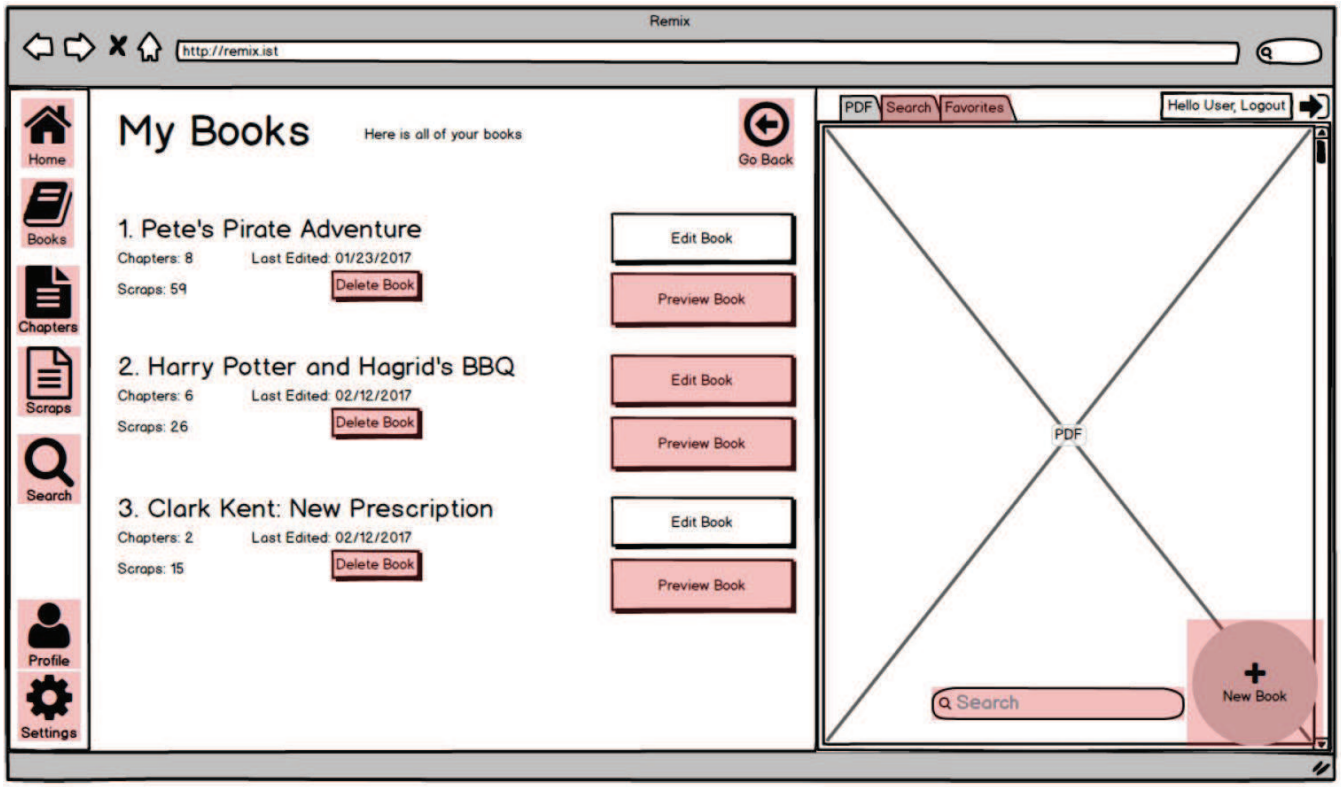


Fig. 2. Here is one medium fidelity prototype, created in balsamiq, that allowed for users to provide feedback about the basic ideas of placement without worrying about finalized plans.

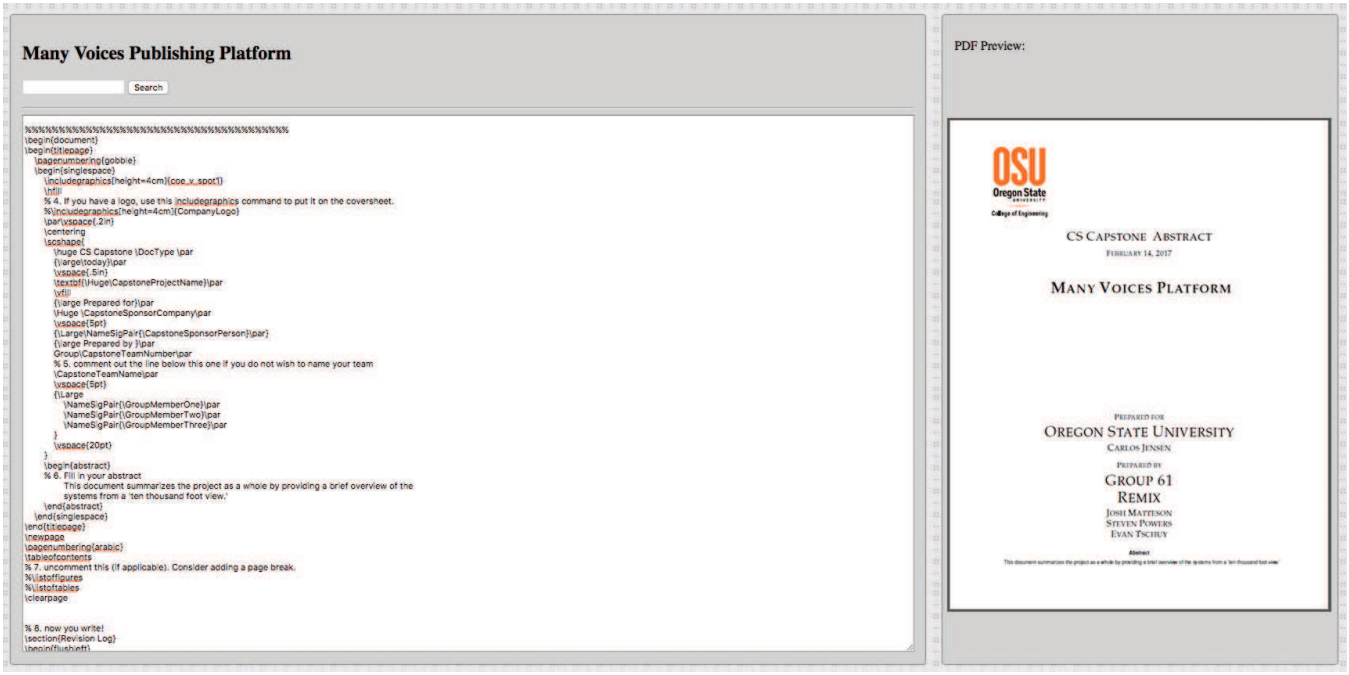


Fig. 3. Here is one high fidelity prototype. This was created using basic web HTML and our Aurelia framework and the use of a hard coded PDF to display a document. This was used as a learning exercise.

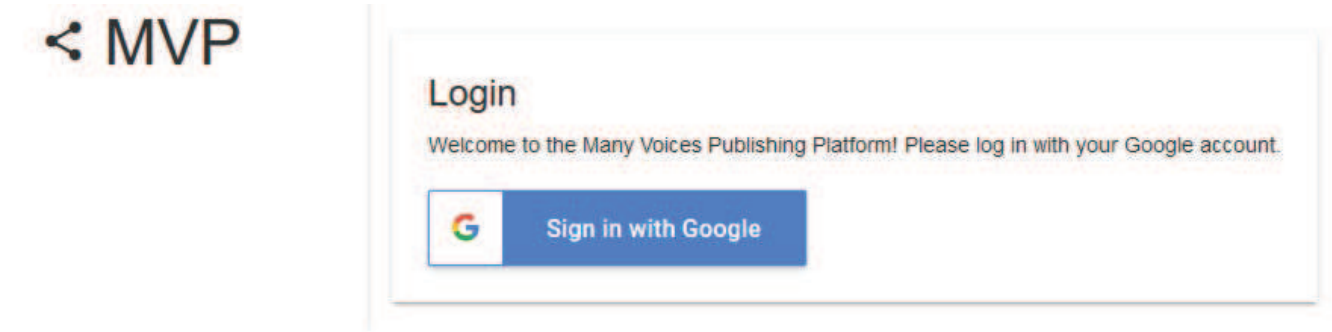


Fig. 4. Here is the final product login page, which uses Google OAuth to abstract our need for security.

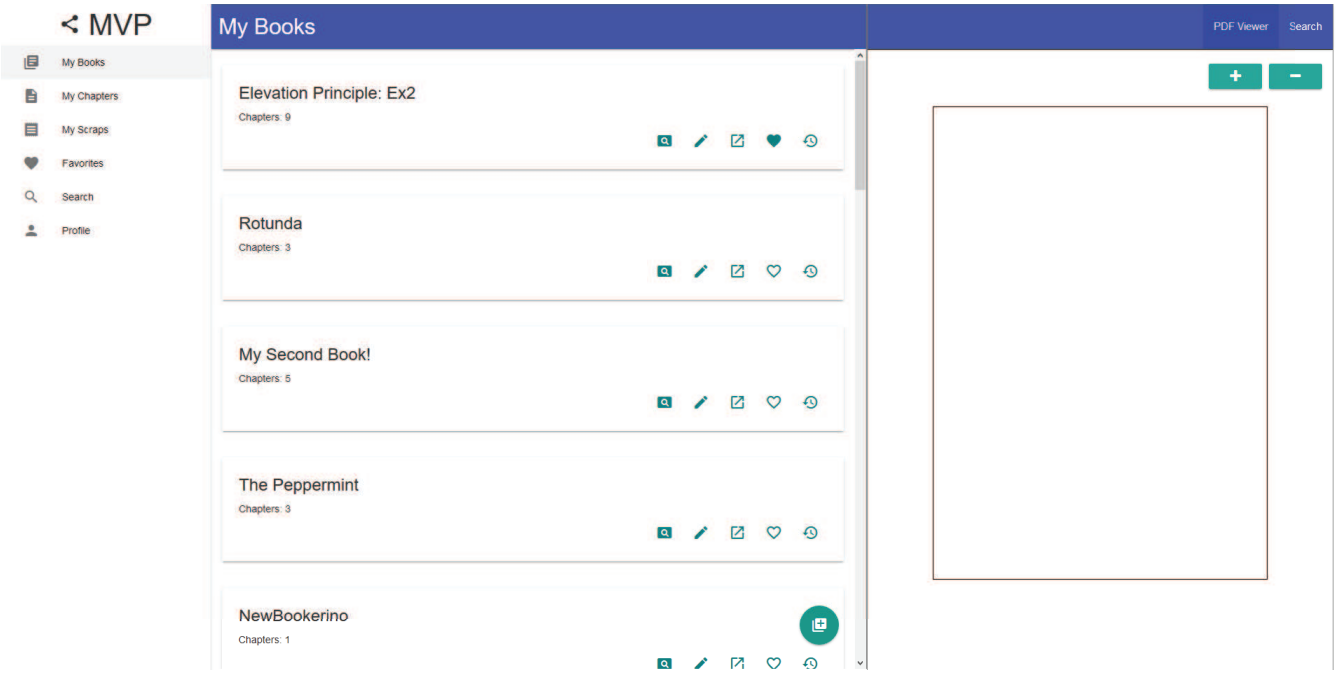


Fig. 5. Here is the final product my books page, showing sample textbook data.

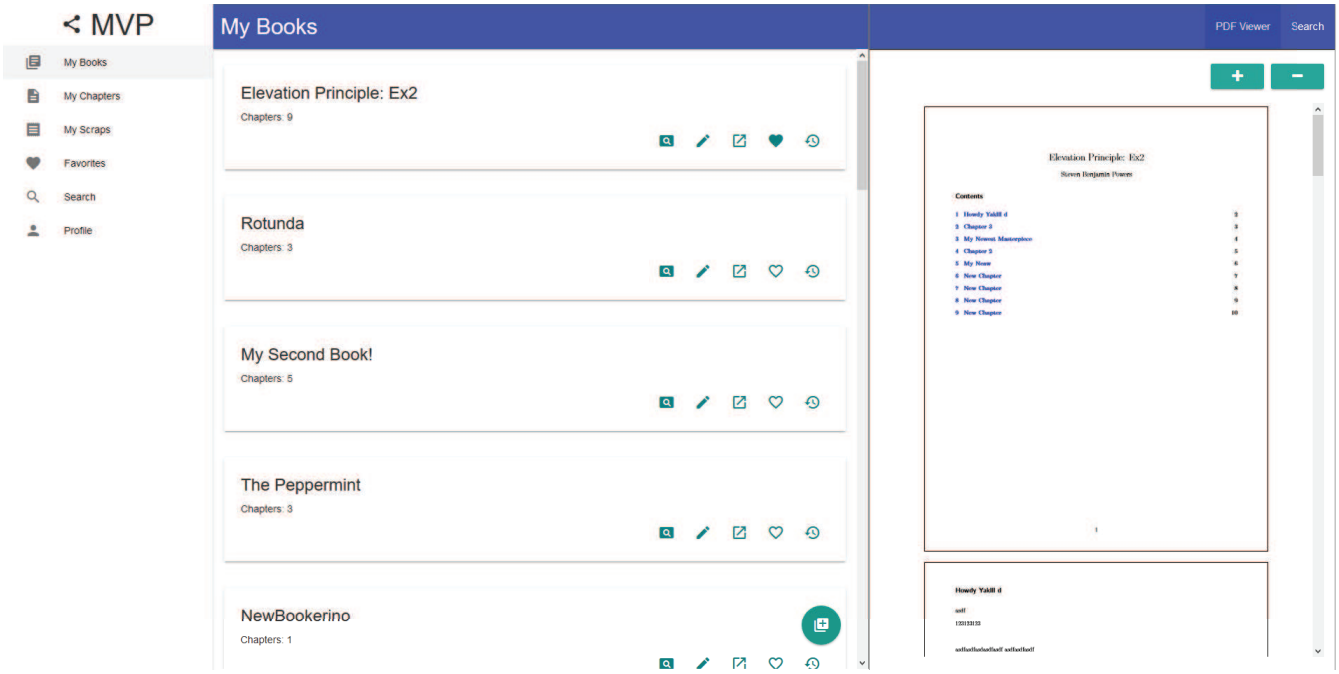


Fig. 6. Here is the final product my books page, previewing a textbook.

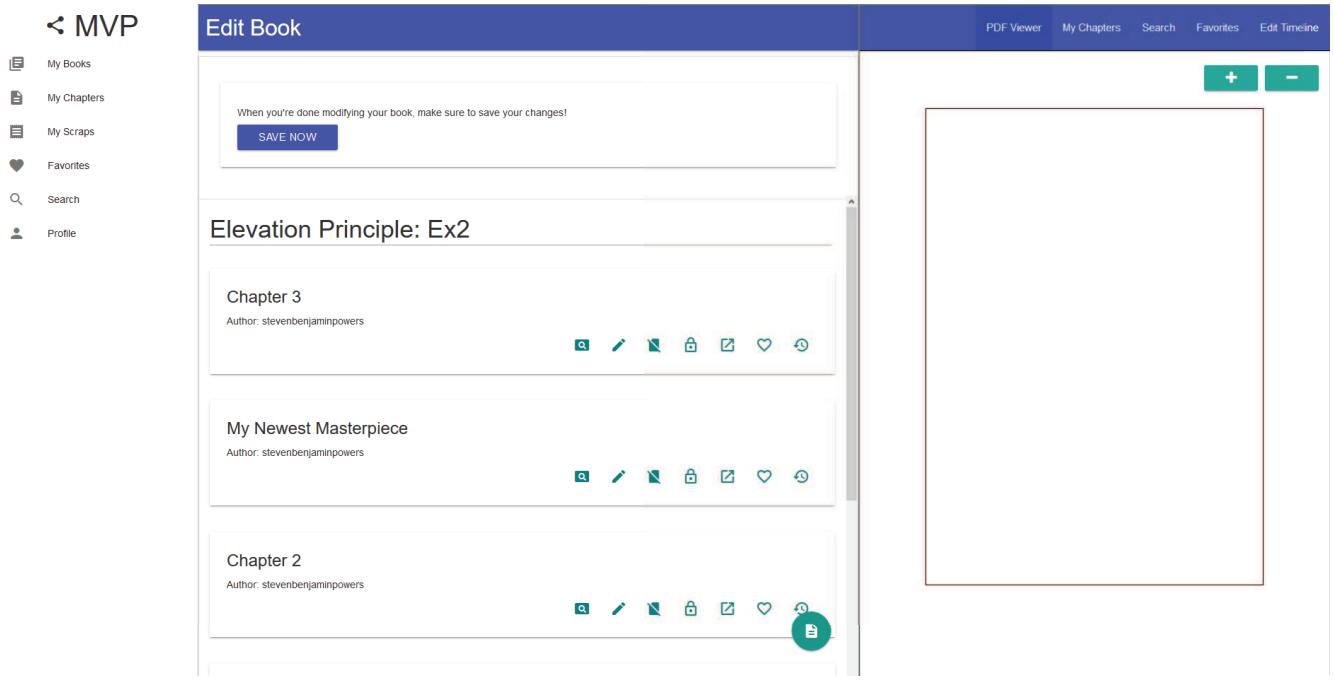


Fig. 7. Here is the final product edit books page, allowing the user to edit chapters within a book.

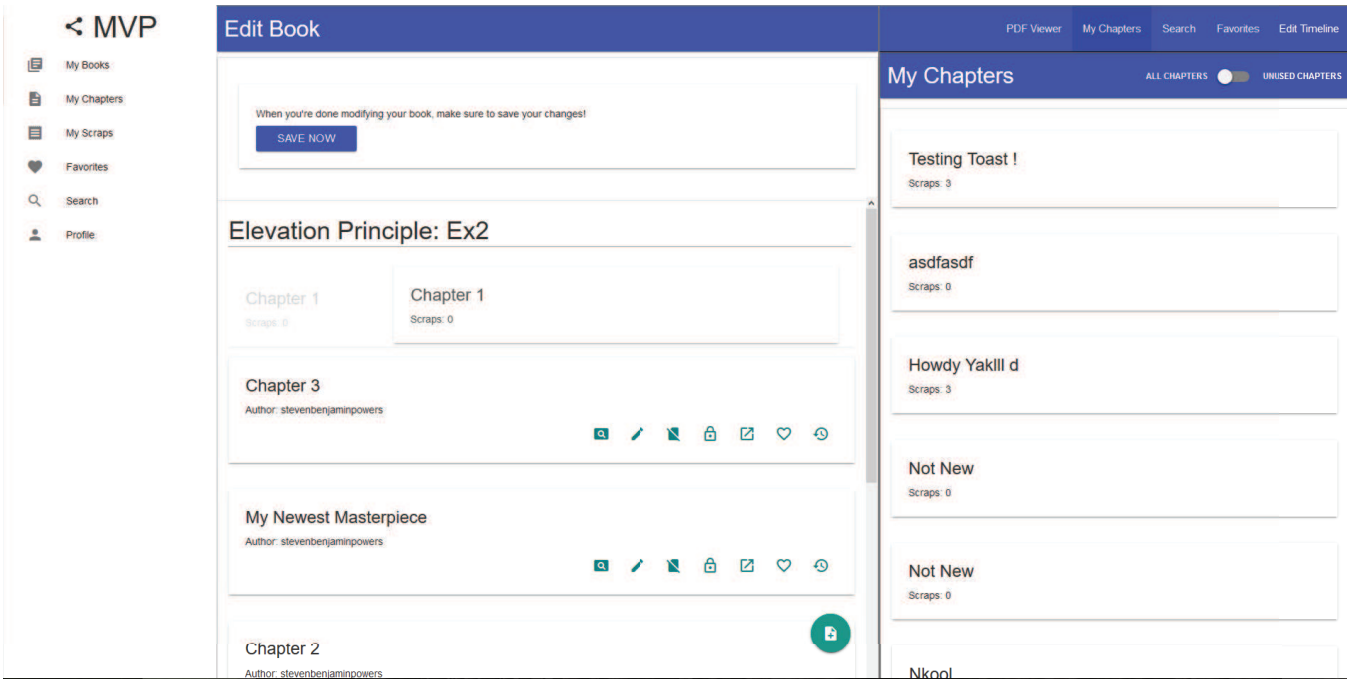


Fig. 8. Here is the final product edit books page, showing the use of drag and drop from a user's chapters.

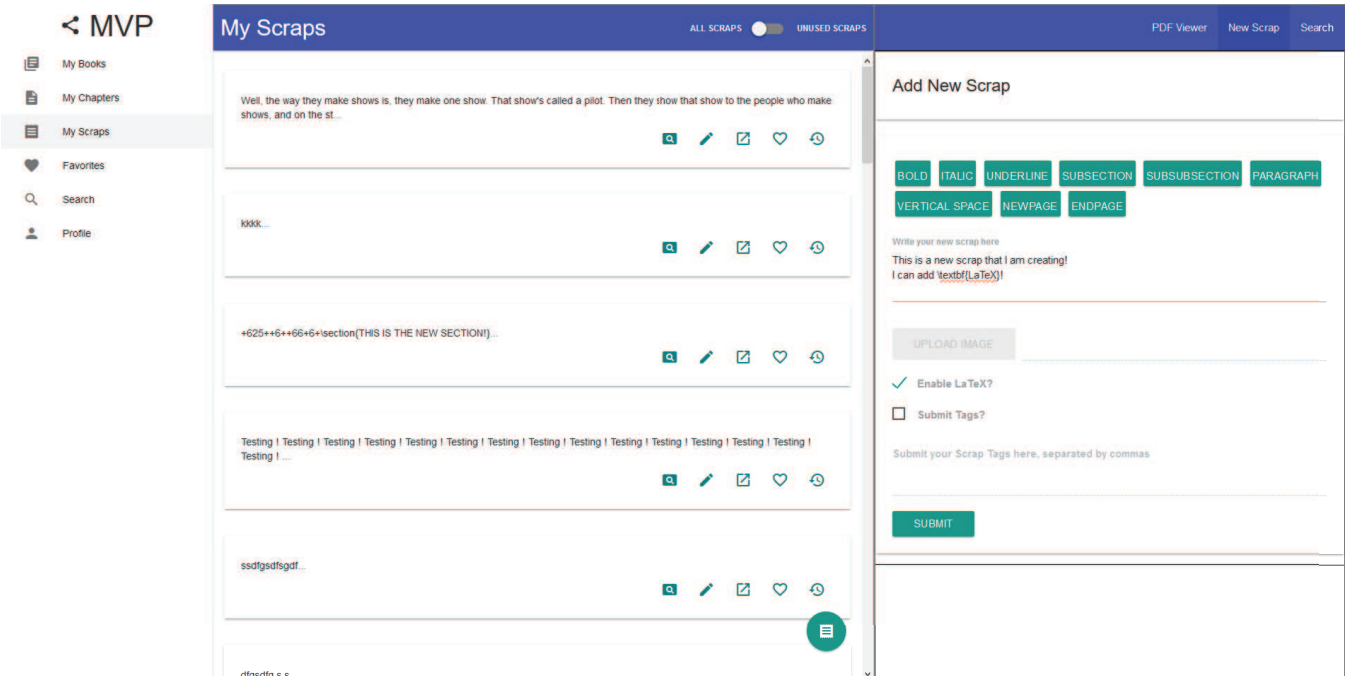


Fig. 9. Here is the final product new scrap page, showing how easy it is for a user to create a new scrap and even enable more advanced features by using LaTeX..