– Independent Work Report: Fall, 2016 –

Bookbag: A Web Application for Collaborative Textbook Authoring and Distribution to Students

Joseph Salter
Adviser: Robert Fish
January 10, 2017

Abstract

The current structure of the college textbook industry places a high burden on both students and professors. Students are faced with either the cost burden of purchasing textbooks in full, or the organizational difficulties that arise from courses that pull readings from a number of different sources. Professors are faced with the task of either finding a textbook that suits their course, writing a textbook themselves, or gathering relevant excerpts from an array of academic material. Bookbag offers a solution that tackles the issues confronted by both parties. On one hand, the student side of Bookbag's platform lowers costs and enables easy organization and customization of academic material. This paper will focus on the other side–the troubles faced by the professor–and offer a potential solution based on easing the path towards collaborative authoring of academic material. Bookbag rethinks textbooks as aggregations of chapters, written collaboratively by any number of authors, created and distributed to students on one platform. I will discuss the product as a whole, but focus mainly on my role in the project: implementing our unique approach to collaborative writing. Bookbag utilizes an abstraction over the web-based Git repository hosting service–Github–to simplify and manage the cooperative writing process for the professor. The result is an intuitive platform that enables professor collaboration on course content, as well as organization of course materials.

1. Introduction

The most apparent issue in the college textbook industry is the extremely high cost that students and their families are faced with each year. There have been numerous studies measuring the average cost of course materials per year for the typical college student, and the results are staggering—the lowest being about $600 [2] and the highest reaching $1,000 [11]. Another study indicates that 65% of college students have opted not to buy a college textbook due to its excessive price at least once during their college experience [10]. As students, we strongly believe that course materials should be easily accessible without a cost burden. This sparked the original motivation behind Bookbag.

After talking to professors, it soon became evident that there was another motivation for Bookbag—the general faculty consensus towards college textbooks was a feeling of dissatisfaction similar to that of the students. Professors often find it difficult to come across a textbook that suits their course, so they are forced to gather content from a variety of sources. The other alternative for the professor is to write a textbook from scratch, but this a large commitment that requires a substantial amount of time, and publisher infringements are often difficult to bypass. As a result, some professors require the purchase of a textbook but end up straying away from it with outside material. Research shows that this often results in professor-student friction—students want to feel that the textbook they purchased was a sound investment, absolutely necessary for the course, but are often frustrated when the course doesn't completely match the textbook's structure and contents [11]. Other professors express dissatisfaction with university, legislative, and publisher actions that place limitations on their freedom to select and replace textbooks [13]. Print textbooks are, after all, limited in how quickly and efficiently they can make changes and correct errors.

Bookbag approaches the college textbook issue from both sides, but its solution is rooted in the idea of collaborative content creation. If the path to writing and editing course material can be eased for the professor, textbooks can be rethought of as groups of dynamically changing content rather than static pieces of text. We seek to give professors the ability to collaboratively generate course materials and distribute them to students all on one platform. This paper will explore the

development of Bookbag's collaborative writing tool, which is a simplified form of version control built on top of Github. I will begin by investigating related work in the field, and move forward to describe our unique approach to and implementation of a collaborative writing platform. Finally, I will close by considering the business aspects of Bookbag as an entrepreneurial venture and offer insight into Bookbag's limitations and future work.

2. Related Work

Part of the motivation behind Bookbag is that there does not currently exist a platform that serves the purpose of both collaborative academic content creation *and* distribution to students as course materials. There do, however, exist implementations of collaborative authoring that successfully accomplish other goals, but fall short in one or more ways to what Bookbag seeks to achieve. It is necessary to explore their approaches to gain insight into the strengths and weaknesses of the existing collaborative methodologies.

2.1. Collaborative Academic Platforms

Perhaps the most well-known online encyclopedia of information is Wikipedia, which allows collaborative content publication to the web. Wikipedia falls under the category of what Berkeley's Robert Glushko describes as Open Collaboration, a process where "a potentially unbounded number of contributors, who are unlikely to know each other, voluntarily create and edit content about some subject for which they claim expertise" [9]. Wikipedia and other Open Collaboration platforms, however, do not verify the validity of the people editing their information, and they lack the two sided professor-student platform that Bookbag seeks to implement.

Google took on the challenge of collaborative writing by implementing a live document editing platform. Google Documents enables multiple users to view and edit a document in real time. This process is great for people trying to collaborate on a quick document together, but it does not share Bookbag's purpose, which is strictly academic. The live editing feature can also get a bit hectic and requires Internet connectivity. Google Docs, further, leaves no freedom for users to select their own file format, a limitation that Bookbag avoids by allowing users to collaborate in whichever

file format they choose, as well as edit files offline. Bookbag also differentiates itself by creating a *classroom* feeling with a professor and a student side, an implementation that Google Docs does not seek to accomplish. ShareLaTeX is a similar platform that allows live collaboration of LaTeX files, but its limitations correspond to those of Google Docs.

Quilt, a collaborative writing brainchild of Bell Labs, provides a framework that addresses the issue of file format limitations by allowing collaborators to use whichever text editors they please. [8]. Bookbag models much of its collaborative writing tool off the implementation goals of Quilt, which recognize and accommodate the variability in tools used for a collaborative project and seek to create a collaborative environment, rather than an editor or window manager. Quilt's goal is to create the framework for collaborative authoring, a goal upon which Bookbag seeks to expand by allowing authors to publish their materials directly to students.

Coursera, an online platform that contains video courses from top universities around the country, encompasses the relationship with students that Bookbag seeks to create. But it lacks the collaborative text authoring component—courses consist of video lectures, rather than text chapters.

2.2. Academic Literature

There does exist an academic discourse on collaborative writing and its potential implementations and limitations. Glushko, in addition to the Open Collaboration concept of Wikipedia, cites Consensus Collaboration, which entails a group of authors reaching an agreement about how and what to write about, and Hierarchical Collaboration, where a single author has a vision for a collaborative piece and manages the contributing authors and writing process in top-down style [9]. Bookbag seeks to combine Consensus and Hierarchical Collaboration, giving special permissions to an owner, but allowing all collaborators to contribute democratically to the work.

The potential success of such a service is visible in the results of a five-year ongoing study conducted at the University of Idaho, where students in an introductory economics course were surveyed about the usage of a custom, professor-written online textbook. While Bookbag does not aim to provide entire online textbooks to students, the study provided an indication of a student

liking to such a resource. The online textbook, according to the students, was a beneficial learning tool and eased a heavy monetary burden for typically expensive economics textbooks [11]. The goal of Bookbag is to allow professors to provide customized course materials to their students in a similar manner, but without requiring them to write entire textbooks.

3. Approach

Bookbag seeks to ease the creation of academic material by rethinking textbooks as aggregations of chapters written by a variety of authors. Our platform allows professors to quickly and collaboratively publish a short work about a single topic, such as the Merge Sort or Cellular Respiration. Professors can then group these texts, which we refer to as "Chapters", into folders, which we refer to as "Courses". Lastly, they can distribute these courses to their students. Figure 1 outlines the typical work flow for a professor on Bookbag. This section focuses on our unique approach to collaborative authoring, which we choose to implement using existing technologies in collaborative software development and version control. We seek to simplify their usability in an attempt to extend their use cases to collaborative text authoring.

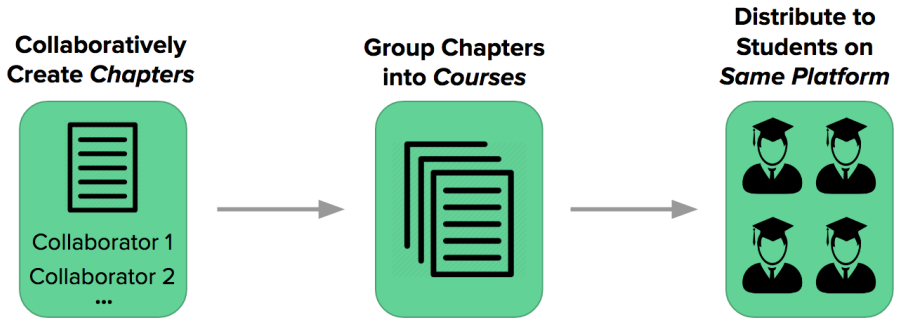

Figure 1: Overview of approach to professor's usage of Bookbag.

3.1. Examination of Existing Technology

Perhaps the most powerful version control tool in use today is Git. Primarily used for software development, Git is a mechanism that keeps track of changes in files. To begin a project with Git, one must create a repository, which can be thought of as a folder on a computer's file system that

has been initialized with a “git init” command. From the point of this initialization onward, any change in the repository can be tracked. To save the current state of files in the repository, a user of Git will, upon making a change to a file/s, perform a “commit” command, which tells Git to essentially take a snapshot of all the files in the repository at their current state. The user can then make another change, perform another commit, and Git will take another snapshot of the updated file state. The user can view the files in the repository at any stage in its history or, in other words, at the time of any commit. Git is most commonly used in software development because it provides an easy and intuitive mechanism for tracking changes in code, finding bugs, adding features, and so on. Git is typically used with Github, a service for hosting Git repositories on-line. By creating remote repositories, Github allows multiple people to contribute to a repository. It is used primarily for collaborative software development, as it provides simple functionality for branching, merging, versioning, and peer code-review.

Despite the fact that Git and Github’s primary use cases involve files containing code, that is not to say that the services do not work for ordinary text files. In fact, they work just as well in their ability to track changes. Why, then, isn’t Git used for collaboratively authoring text context, in addition to its primary purpose of software? Well, because academics and researchers in non-technical fields are either unaware of the existence of Git or intimidated by its technicality. But, as the most powerful tool for collaborative version control, we want to extend Git and its powerful partner, Github, to all reaches of collaborative authoring.

3.2. An Abstraction Over Github

Bookbag, then, creates an abstraction over Github that provides all of its main benefits without requiring any knowledge of how it technically works. While assuming a knowledge of Git could have made it easier to include more of its features, like branching and merging, it would limit our target professors to those in computer science or other technical fields. Bookbag allows for both collaboration and version control at the same level of power that Github provides, but with a completely separate interface and terminology.

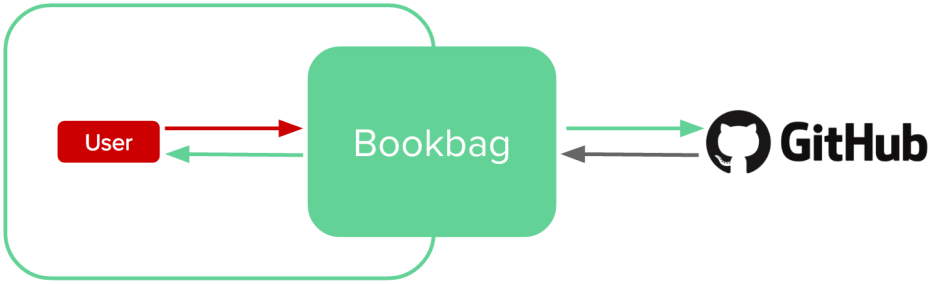

Figure 2: Bookbag's interaction with Github completely hidden from the user.

As seen in Figure 2, a user of Bookbag will communicate only with Bookbag and Bookbag will communicate back to the user. Bookbag's underlying interaction with Github, which is the foundation upon which Bookbag's collaborative authoring component works, is completely hidden to the user.

3.3. Key Features

Table 1 displays the features that a Bookbag professor will be able to utilize compared directly to their equivalent actions on Github.

Bookbag Feature	**Github Action**
Start new chapter, upload initial work	Create new repo with intial commit
Add collaborators to a chapter	Add contributors to a repository
View current or previous version of chapter	Open current files in repo or view old commit
Edit chapter contents	Change files, commit, push
Publish chapter, make available to students	Make repository public

Table 1: Bookbag features with their equivalent actions on Github.

Professors, upon logging in, will be able to either start a new chapter, making them the "owner" of that chapter, or make edits to a chapter that they own or are contributing to. A chapter owner will be able to add collaborators to the chapter, giving other professors permission to make edits. How a group of collaborators chooses to build their chapter is determined completely by them. A chapter, for example, could include a source file (e.g. Microsoft Word, LaTeX) and a PDF version, or a number of different source files that will be combined into one PDF later on. All of the files contained by a specific chapter will be able to be viewed and edited by all contributors to that chapter. When the owner of the chapter decides that it is complete, s/he will be able to upload a

final PDF of the chapter and make it public. The chapter will now appear in the search results if a professor or student searches Bookbag for that specific topic or author, and it will also be available for any professor to add to a course. At this point, despite the chapter being public, contributors will still be able to edit and update it privately. The chapter owner can change the public version at any time.

3.4. The Library Book Metaphor: "Checking Out"

Bookbag's collaborative writing process is most easily described by equating each chapter to a dynamically written library book. To make an edit to the library book, a professor must *check out* the book. Only one professor can have the book checked out at a time, and within x number of hours (x is set by the book's owner), they can make edits and *check the book back in* with its new changes. Users that do not have the book checked out cannot make edits to the book, though they can still view its contents. Only the sole professor that has the book checked out can actually check-in changes. Any professor can make their own edits on the side, but these will only become official if they are inserted to the book during a checkout. Each time a chapter is checked in, a new *version* of the chapter is saved. Every version of the chapter can be viewed at any time.

3.5. Benefits of This Approach

Below is an enumeration of the main advantages of our approach to collaborative authoring:

- Abstraction requires no prior knowledge of version control, Git, or Github
- Checkout concept removes the issue of merge conflicts and multiple people trying to make edits at one time
- Easy to view or revert to old versions
- Collaborators can build their chapter in whatever format and however many files they please

4. Implementation

Bookbag's backend consists of a server built with Node.js, using the Express.js web framework, hosted on Heroku. All data is stored in a PostgreSQL database. The entire code base is available

at the Github link in Appendix A. To implement the abstraction over Git, I created a module tools/Git.js that communicates with two things: Github, using its public API, and our database. In this section I will delve into how our collaborative writing tool is implemented, including the specifics of which tasks require Github interaction and which require database interaction. Figure 3 lays out the architecture of the entire backend, portraying how exactly the Git abstraction fits in. The server contains API calls that communicate directly with the Git abstraction, requesting or sending information about specific chapters. The Git abstraction then makes the appropriate call/s to Github or the database, depending on the action, and returns information to the server.

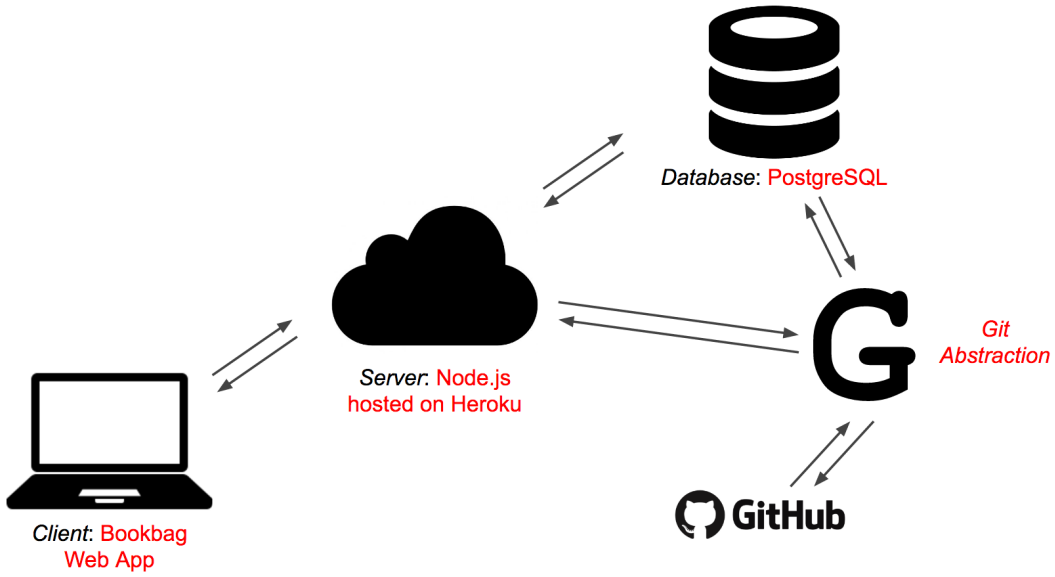

Figure 3: Backend architecture of Bookbag.

4.1. Github's Role

Github plays the largest role in the Git abstraction. Its job is to hold the actual content of each chapter—all files that make up each chapter, as well as the history of each chapter. I created a company account on Github, under the username bookbagInc, that I am the manager of. Each chapter on Bookbag is its own repository on bookbagInc's Github page[1]. To interact with this account and its repositories in the Git module, I make use of a Node.js module called node-github

[1]The account is not yet private because this requires a paid Github subscription. When the app is actually deployed, bookbagInc will surely be a private Github account to protect all of the chapter information.

[3], which is a Node.js wrapper for Github's API. Each request to the API requires authorization with a secret API key; this ensures the security of the content in bookbagInc's repositories.

The simplest example of Github's usage occurs when a professor indicates that they would like to begin a new chapter. In this situation, the following steps would occur:

1. The frontend communicates with the server, using its /api/prof/createnewchapter call
2. The server makes a call to the Git module's createNewRepo() function, passing as a parameter a hashed ID of the desired name of the chapter
3. Within this function, the Git module uses the node-github wrapper to make a POST request to Github's /user/repos API, creating a repository for the chapter on Github under the account bookbagInc
4. The Git module returns chapter creation success or failure to the server, which passes the information along to the frontend
5. Upon success, the professor will be able to work on the new chapter, add collaborators, etc. The underlying Github process is completely hidden

Table 2 delineates all frontend chapter-related actions (except those dealing with editing; we will talk later about this) and their respective calls to the server. Table 3 similarly depicts what is happening on the backend—the Git module function associated with each of the professor actions and their respective Github API calls. (Row 1 of Table 2 corresponds to row 1 of Table 3, and so on).

Professor Action	**Bookbag API Call**
1. create new chapter	1. /api/prof/createnewchapter
2. delete chapter	2. /api/prof/deletechapter
3. view history of chapter	3. /api/prof/getchapterhistory
4. download most recent contents	4. /api/prof/getchaptercontents
5. download contents for previous version	5. /api/prof/getchaptercontentsprevious

Table 2: Professor actions and their respective Bookbag API calls.

You may notice that actions 4 and 5 in Table 3 use the same Github API call; action 5 simply

Git Module Function	Github API Call
1. createNewRepo()	1. POST /user/repos
2. deleteRepo()	2. DELETE /repos/:owner/:repo
3. listCommitsForRepo()	3. GET /repos/:owner/:repo/commits
4. getLatestContentsOfRepo()	4. GET /repos/:owner/:repo/contents/:path
5. getContentsOfRepoForCommit()	5. GET /repos/:owner/:repo/contents/:path

Table 3: Git module functions and their respective Github API calls.

provides another parameter specifying the unique identifier of a previous commit, referred to on Github as the SHA, or hash, for that commit.

Editing the contents of a chapter is bit more complicated. To make an edit, a professor, who must have the chapter checked out, uploads the edited file/s on the frontend and provides a short note (commit message) explaining what has been edited. The frontend will then make a call to Bookbag's /api/prof/upload API. Upon making this call, the server asynchronously calls the Git module function makeBlobForRepo() for each file that the professor uploaded. A blob is simply an encoded file that can be referenced with a unique SHA. Once a blob is created for each file using Github's blob API, the server then calls the Git module's createCommitWithBlobArray() function, which, as its name indicates, creates a new commit with the array of blobs that were just created. Doing so requires five consecutive Github API calls, all of which only occur if the previous call returns success. This function takes the following steps:

1. Retrieve the SHA of the most recent commit; this will become the "parent" of the new commit that we are creating
2. Retrieve the tree object of the most recent commit. All commits on Github contain a tree object that holds all of the data of the current state of the repository.
3. Create a new tree that adds the new blobs onto the tree that we retrieved in Step 2.
4. Create a new commit with the parent retrieved in Step 1 and the new tree that we created in Step 3, which contains the previous state of the repository plus the new files uploaded
5. Update the current reference of the master branch to point to this new commit

Editing a chapter, then, requires a file/s upload, which, on the backend, triggers a commit to

the chapter's Github repository. The edit will now appear in the chapter's history. Step 3 contains perhaps the most powerful feature of Bookbag's collaborative writing tool. Because we are *adding on* to the current tree, all the existing files in the repository will be preserved in the new commit. If a new file has the same name as a previous file, the previous file will be overwritten. But the magic of Github allows the previous file to remain accessible in a previous commit. Therefore, the professor can view any file at any stage in the history of the chapter.

A professor can also revert the contents of the chapter back to a previous version, or a previous commit in the repository's history, if s/he is the owner of that chapter. This process is similar to the one just described. It requires the retrieval of the parent SHA, the retrieval of the tree object that we wish to revert back to, the creation of a new commit with the old tree and parent, and an update of the master reference to point to this new commit. This feature was built in the Git module, but has not yet been implemented on the frontend.

4.1.1. Challenge: Tracking Author of Each Commit The assumed etiquette for collaboration states that contributors must know *who* is making edits and *where* they are making them. Professors on Bookbag, then, should be able to see the author of every commit, or every edit, of the chapter. But, on Github, the author of each commit is technically bookbagInc—the account that manages the entire abstraction. This posed a challenge, which I solved in the following way:

1. Upon a file/s upload to a chapter, retrieve from the server the name of the user that performed the upload
2. When creating the new commit to send to Github, craft the commit message string in the following way: "[Author] + **** + [commitMessage]", where commitMessage is the user provided string explaining their file edits
3. Then, when listing the commits of the repository, split the string with the separator "****", and return both the author and the commit message for the frontend to display

Therefore, the author of each commit is actually contained in the commit message on Github, though it appears separately to the Bookbag user.

4.1.2. Error Handling Because some of the Git functionality requires upwards of four to five calls to Github's API, handling errors on each call is critical to preventing server crashes. Node-github simplifies the error-checking process—each call to the Github returns an "error" and "response". If there is an error, I am able to return failure to the server with an error message. If not, I can perform the next call to Github, which typically requires some information stored in the response. This way, each consecutive call is asynchronous and only occurs if the previous call was successful.

4.2. Database's Role

As Github controls chapter history and content, the database controls chapter collaborators as well as the checkout functionality. The database contains a "chapters" table, which holds information *about* the chapters, *not* their contents. In addition to the chapter name and owner, each chapter object in the database has the following fields that are essential to the Git abstraction: contributors, checkout_user, checkout_expiration, checkout_duration, public, and pdf_url. The owner of a chapter can add or remove from the contributors list; this happens with database queries. A professor will only be able to edit a chapter if they are in its list of contributors. The checkout_user field contains the unique ID of the professor that has the chapter checked out at that given time. The field is empty if the chapter is not checked out at the time. The checkout_expiration field holds the date and time at which the current checkout session will automatically expire, and the checkout_duration field holds the maximum length of each checkout session. The default checkout duration is three hours, but a chapter owner can customize this upon creating a chapter if a longer or shorter checkout period is desired. A professor, upon uploading edits to a chapter, can check the chapter back in, expiring the checkout session early.

4.2.1. Making a Chapter Public When the owner of a chapter decides that it is ready to publish, or make available to students and other professors, they are required to select one of the existing PDF files in the repository to become the public version. The file they select can originate from any commit in the chapter's history. Upon selection, that chapter's "public" flag in the database is set to true, and its pdf_url field is set to the download URL of the file that was selected. This URL

is retrieved from Github. At any point, an owner can change the public version to a different PDF within the chapter.

4.2.2. Database Security I also had a small role in creating Bookbag's security protocols. Specifically, I wrote a module /tools/auth/Hash.js that utilizes a Node.js library called bcrypt-nodejs [1] to hash passwords on account creation and to verify passwords on user login. Our database is therefore storing hashes of user passwords, a simple but necessary security component of any web application. My initial goal was to utilize Passport [4], a user authentication middleware for Node.js, and I built the architecture to make this happen in auth/config and auth/models. Passport would have increased the database security of our web application, but Bookbag's collaborative writing platform took precedence and forced Passport to fall to the back-burner. Hashing passwords makes the database secure enough in Bookbag's current version, but the Passport modules I wrote should eventually be incorporated for an extra layer of security.

4.3. Professor Verification

A requirement of Bookbag is that its academic content is credible and trustworthy. While we are not editing and verifying the chapters that professors write, we want to ensure that the people who are writing them are qualified in the first place. There does not exist a database of every qualified professor, lecturer, and researcher in the country. As a result, it seems that we would have to do a unique university faculty search of anyone who attempts to create a professor account on Bookbag, based on the institution they are associated with. This, given different searching conventions for different university websites, was deemed too difficult and not completely necessary for Bookbag's current goal of gaining credibility at Princeton before expanding to other schools. As a result, I wrote a module tools/auth/ptonVerify.js that verifies a potential Princeton faculty member against the titles "professor", "lecturer", or "researcher". To do so, I perform a search of Princeton's Advanced Directory Search [5], requiring one of these three keywords. Account creation will fail if the attempted user's title does not contain one of them, or if the provided first name, last name, and email do not return any search results.

5. Results

It is perhaps easiest to understand the final product of the Git abstraction by examining the frontend user interface, for which I helped design the layout with Jake Levin, our frontend engineer. Appendix B contains the link of the application deployed on Heroku. Figure 4 displays a dummy chapter about Binary Search, written collaboratively by two users that we created, Professors Kevin Wayne and Robert Fish. Robert Fish is the chapter owner and, given the planet icon on the top right of the screen, we know that he has published one of the PDF files in the chapter. Moving down from the planet icon, we also know that we are logged in currently as Dr. Fish because we have the ability to "Update Published Version" (this button would simply say "Publish" if the chapter has not yet been published). Only the chapter owner has this capability. Hitting this button will present a dropdown with a list of only the PDF files of the chapter version that the professor is currently viewing, one of which will be selected to make public. In this example, that dropdown would allow the professor to publish either "binary_search_final.pdf" or "binary_search_inprogress.pdf".

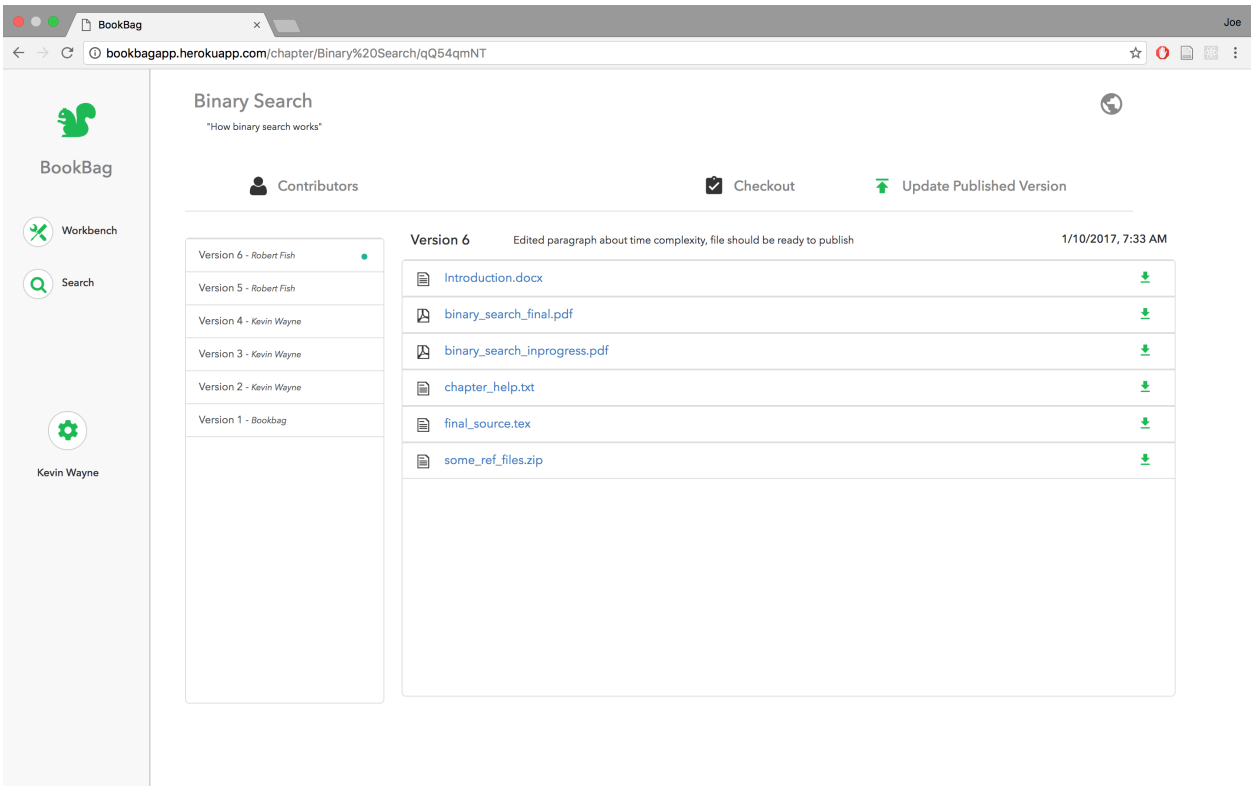

Figure 4: User interface over the Git Abstraction.

Under the chapter name and description, we see a “Contributors” button, which on click presents a dropdown list of everyone contributing to the chapter. The chapter owner will have the ability to add or remove contributors from this list.

We can also see a “Checkout” button, which enables the logged-in professor to check the chapter out. Upon checking out, the “Checkout” button will now say “Upload Files and Check In”, which will allow the professor to upload files to the chapter with a commit message (see Figure 5). Upon hitting the “Upload” button, the changes made will appear as “Version 7” in Figure 4. The author of the upload will be specified next to the version number.

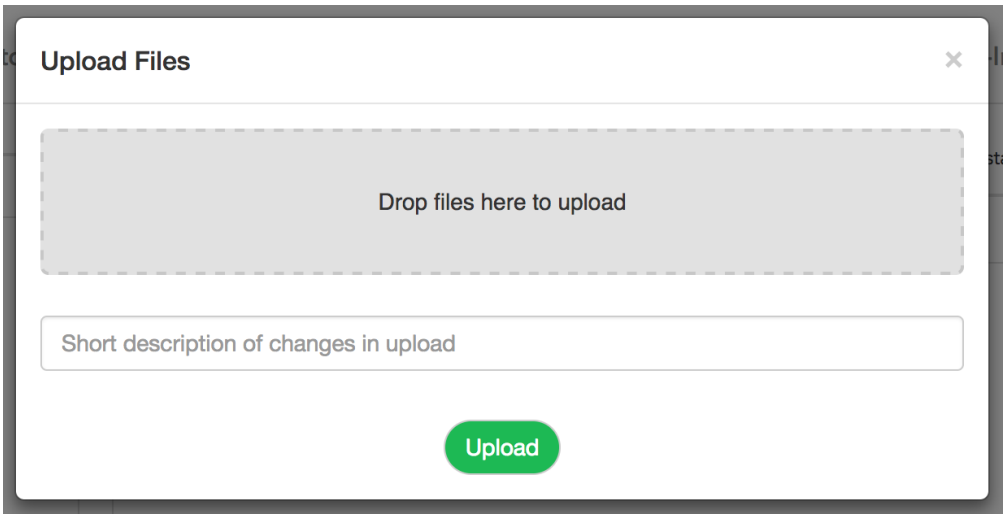

Figure 5: Upload files popup.

The file list—the contents of the chapter—take up the majority of the page, and represent the files at the version the professor is currently viewing (specified by the green dot next to “Version 6 - Robert Fish”). Clicking on a file will download it. Above the file list is the version number, the short message from the author of that version, and the date/time of the edit. The professor can switch from version to version, viewing the files at any stage of the chapter.

All of the features described work seamlessly, performing the necessary backend actions with Github or the database and returning information to the client.

6. Entrepreneurial Considerations

Bookbag is currently in a beta stage, the primary goal of which is to recruit Princeton professors on board and get content uploaded to the site. It does not make sense to monetize the product until it is fully usable, with a substantial amount of professors committed to using Bookbag for their courses. However, given the entrepreneurial spirit of Bookbag and its underlying motivations, it is necessary to discuss potential business strategies for monetization. After all, a primary motivation for Bookbag is to reward professors for their knowledge by paying them for the content they publish on the site.

6.1. Business Model

It was decided to model the revenue and payment strategy most closely off of Spotify's. Spotify, the most popular music-streaming service, receives some revenue from advertisements, but a vast majority of the money they bring in is from user subscriptions. To pay an artist, Spotify multiplies their total monthly revenue by the fraction of streams said artist accounted for. After taking a cut of the result for themselves, Spotify pays the artist, making the artist's profit proportional to its number of streams divided by the total number of streams [12].

Bookbag's model is similar—professors and their chapters are analogous to artists and their songs. Students would pay a semesterly subscription to Bookbag, and each chapter that they subscribe to equates to streaming a song on Spotify. The complication with this model is that each chapter will likely have a number of collaborators. So, we can pay each *chapter* by multiplying the total semesterly revenue by the percentage of subscriptions that chapter accounted for (after taking a small cut for ourselves). How to pay each *professor* is a bit more confusing. To divvy up the chapter's revenue amongst its collaborators, we considered paying each collaborator proportional to the number of edits they uploaded, or the number of words they wrote. We, however, feared that this may cause unnecessary edits or verbose language for the sake of earning more revenue. As a result, we decided that the best model would be to allow collaborators of a chapter to discuss amongst themselves the proper splitting of the revenues, and decide the percentage that each collaborator

should receive. This is analogous to paying a band on Spotify—the members of the band decide for themselves the splitting of their total profit.

6.2. Profit Potential

This section explores the profit potential for professors by experimenting with incrementing numbers of student subscriptions and subscription costs. The table is phasal, with Phase 1 representing Bookbag's goal at Princeton, Phase 4 representing a substantial infiltration into the market, and Phases 2 and 3 representing incremental expansion in between. The table denotes the average profit a professor will receive in each phase, with student subscription costs ranging from $75 to $125 and an average of three collaborators per chapter. For the sake of the example, each chapter's revenue is divided evenly amongst the three professors. The profit results take into account a 10% cut collected by Bookbag.

	Phase 1	**Phase 2**	**Phase 3**	**Phase 4**
# Students subscribed	2,500	50,000	250,000	1M
# Professors contributing	100	1,000	5,000	10,000
# Chapters on site	200	1,000	2,500	5,000
# Avg. Chapters/Student	30	40	50	60
Profit / Semester / Chapter:				
- Subscription cost: $75	$80	$1,125	$2,250	$4,500
- Subscription cost: $100	$375	$1,500	$3,000	$6,000
- Subscription cost: $125	$470	$1,875	$3,750	$7,500

Table 4: Bookbag's profit potential for professors.

While the potential for profit in Phases 1 and 2 appears discouraging, the structure of the education system lends to the fact that there are many more students than professors—precisely, about 20.5 million students[2] to 1.3 million professors[3]. As a result, we suspect that the number of students onboard will grow to a substantially higher number than the number of professors onboard. Logically, if a professor decides to try Bookbag for the first time at a new school for a course of 200 students, the number of student subscriptions will rise by 200 while the number of professors

[2] National Center for Education Statistics
[3] Bureau of Labor Statistics

only by one. Potential profit looks promising in Phases 3 and 4, especially if we remember that these numbers reflect profit per semester per chapter. If we extrapolate the data to a full year and, for the sake of example, say a professor is contributing to four chapters, profit numbers would range from $18,000-$30,000 in Phase 3 and $36,000-$60,000 in Phase 4, depending on the subscription price. Our hope is that these numbers are enticing to professors and provide incentive for them to use Bookbag for their courses and publish their own content.

6.3. Market Restructuring

Figure 6 displays the current structure of the college textbook industry. Publishers are at the top, providing an array of potential textbooks to professors. Professors must then decide which textbook to assign to their students, who then pay the publisher for the textbook. Bookbag seeks to restructure the nearly $5.5 billion dollar industry [6] by teaming up with professors to remove textbook publishers from the equation. Figure 7 depicts the flow of content from the professor to Bookbag, through to its distribution to students. Students pay a subscription fee in return. Bookbag then, after taking a cut for itself, relays the revenue back to the professor for the work that they've done.

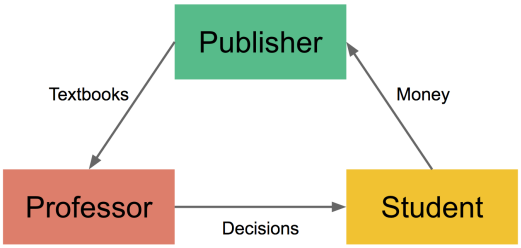

Figure 6: Current structure of textbook market.

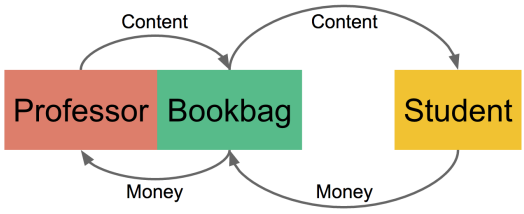

Figure 7: Bookbag's market infiltration.

7. Conclusion

Bookbag, then, is an educational and entrepreneurial venture that uses existing version control technology to change the control mechanism for collaborative authoring. It is unique in the abstraction it creates over Github, as well as the coupling of course content creation with distribution to students. Bookbag addresses the college textbook cost problem, as well as the difficulty faced

by professors of finding a textbook or aggregating readings, by shifting the underlying nature of the textbook from a stagnant text object to a dynamically changing aggregation of course materials. Professors can add to their course, remove from their course, and change the content of the chapters that make up their course as their teaching habits evolve from day to day, week to week, or year to year. No longer must a professor be limited to selecting a textbook and adapting to its contents. With Bookbag, professors can customize and create their own course materials with relative ease.

My primary role in the project was to build the backend foundation on which the collaborative authoring feature of Bookbag rests, as well as some other verification and security components of the application. Being my first experience with web development, building Bookbag with my team required an intensive learning process, as well as strong communication skills, decision-making strategies, and interdependence within the group. It was a valuable experience and we hope to continue Bookbag's evolution through to the academic purpose that it has the potential of filling at Princeton and beyond. The current version of Bookbag is usable and includes all of the main features that we sought to implement, but there is extensive room for user feedback and additional features. The next section describes some add-ons that would enhance Bookbag's usability for the professor.

7.1. Future Work

There are an array of possible features that we hope to implement into Bookbag's collaborative writing tool, the first being an on-site messaging system. This would act as a chat room that would allow contributors of a chapter to discuss amongst each other whatever they please—the plan for their chapter or how they are going to tackle the collaborative process. In the same vein, the current version of Bookbag has no mechanism for professors to provide feedback to each other about edits that were just made to a chapter. While this could be incorporated into the chat room, we hope to provide an on-site file annotation feature that would enable professors to view a file and annotate it directly on the chapter's collaboration page. Both of these features would instill a sense of community discussion and feedback around the collaboration process, an element that is currently

missing from Bookbag. Further, a recent study showed that authors working on versioning of text files most prefer side-by-side comparisons of two documents that clearly display the differences between them [7]. Such a version comparison feature would help Bookbag users gain more insight into the evolution of their chapters from version to version.

Beyond adding features to the collaboration tool and improving the general smoothness and functionality of the application—an ongoing process for every platform—our main goal is to establish Bookbag at Princeton by reaching out to professors and encouraging them to try it for their course. We will take feedback seriously and continue to improve the platform in hopes of making Bookbag the desired platform for course content creation and distribution.

8. Acknowledgements

I would like to thank my group members, Chris Giglio and Jake Levin, for helping turn Bookbag into a reality. I'd also like to thank Professor Robert Fish for his support and guidance over the course of the semester. Lastly, I'd like to thank the Massachusetts Institute of Technology for licensing and creating node-github, which eased the use of Github's API for our application.

9. Honor Code

I pledge my honor that this paper represents my own work in accordance with Princeton University regulations.

x *Joseph Salter*

References

[1] "bcrypt-nodejs." [Online]. Available: https://www.npmjs.com/package/bcrypt-nodejs

[2] Infographic: Course materials - student spending and preferences. The National Association of College Stores. [Online]. Available: https://www.nacs.org/advocacynewsmedia/StudentSpendingInfographics.aspx

[3] "node-github." [Online]. Available: https://github.com/mikedeboer/node-github

[4] "Passport." [Online]. Available: http://passportjs.org/

[5] Princeton advanced directory search. [Online]. Available: http://search.princeton.edu/

[6] D. D. Bramhall, "A short take on: value in textbooks," *The Community College Enterprise*, vol. 15, no. 1, pp. 39–44, Spring 2009, copyright - Copyright Schoolcraft College Spring 2009; Document feature - Tables; ; Last updated - 2010-06-09. Available: http://search.proquest.com/docview/218821244?accountid=13314

[7] D. Dadgari and W. Stuerzlinger, *New Techniques for Merging Text Versions*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 331–340. Available: http://dx.doi.org/10.1007/978-3-642-21605-3_37

[8] R. S. Fish, R. E. Kraut, and M. D. P. Leland, “Quilt: A collaborative tool for cooperative writing,” in *Proceedings of the ACM SIGOIS and IEEECS TC-OA 1988 Conference on Office Information Systems*, ser. COCS ’88. New York, NY, USA: ACM, 1988, pp. 30–37. Available: http://doi.acm.org/10.1145/45410.45414

[9] R. J. Glushko, “Collaborative authoring, evolution, and personalization for a "transdisciplinary" textbook,” in *Companion to the Proceedings of the 11th International Symposium on Open Collaboration*, ser. OpenSym ’15. New York, NY, USA: ACM, 2015, pp. 10:1–10:10. Available: http://doi.acm.org/10.1145/2789853.2789867

[10] T. Kingkade, “Majority of students have skipped buying a college textbook because they’re too expensive,” Jan 2014. Available: http://www.huffingtonpost.com/2014/01/27/textbooks-prices_n_4675776.html

[11] J. R. Miller and L. Baker-Eveleth, “Methods of use of an online economics textbook,” *American Journal of Business Education*, vol. 3, no. 11, pp. 39–43, 11 2010, copyright - Copyright Clute Institute for Academic Research Nov 2010; Document feature - Tables; ; Last updated - 2010-12-18. Available: http://search.proquest.com/docview/818558679?accountid=13314

[12] M. Pollock, “7 things you didn’t realize are happening every time you stream a song on spotify,” Oct 2015. Available: https://mic.com/articles/110690/7-things-you-didn-t-realize-are-happening-every-time-you-stream-a-song-on-spotify#.wgLvf5MUu

[13] L. S. Silver, R. E. Stevens, and K. E. Clow, “Marketing professors’ perspectives on the cost of college textbooks: A pilot study,” *Journal of Education for Business*, vol. 87, no. 1, pp. 1–6, 2012. Available: http://dx.doi.org/10.1080/08832323.2010.542503

10. Appendix

10.1. Appendix A

Bookbag web app code base (front end and back end): https://github.com/jkvlevin/bookbag

10.2. Appendix B

Bookbag client, deployed on Heroku: http://bookbagapp.herokuapp.com/