Book Bags

UI DESIGN FINAL PROJECT

DEVELOPMENT DOCUMENT

THE FEINEST (GROUP 8)

LAUREN ZOU (ljz2112)
KRISTEN HOWARD (kbh2120)
SARAH GREEN (stg2117)
TOM SEGARRA (tas2172)

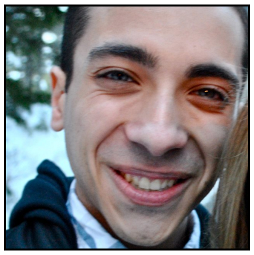
OVERALL PROCESS

When our group first met, we did an extensive brainstorming session in order to define the general concept for our application. We wanted to ensure that the New York Times was visible and essential to our app, while using the NYT API in an unexpected and innovative manner. We also wanted to create a product with real-world relevance beyond this course, which could join the ranks of professional applications using the NYT APIs. We began by choosing to use the Books API (including both Book Reviews and Best Sellers request types), which we felt would allow us to create an application that was not unoriginal or “obvious,” but still connected to print media and the New York Times. We observed an unoccupied niche in the digital ecosystem that our application could fill: a space for users to create lists of books (“Book Bags”), which our application analyzes in order to recommend similar books the user might enjoy. The NYT Books API adds a layer of quality assurance by prioritizing NYT Bestsellers among the recommended books, and by providing direct access to the NYT review for each book.

We started imagining our application as a “Pandora for books.” Pandora Internet Radio is a music streaming and recommendation service that allows users to create “stations” for music they like, and then stream playlists of similar music that Pandora’s algorithms predict the user will enjoy. The user can continuously respond to tracks that Pandora selects, marking it as thumbs-up or thumbs-down, which makes the algorithm learn continuously more about the user’s tastes. Thus it can make increasingly better choices and give the user increasingly accurate recommendations for new music to try out. Pandora also provides helpful links to purchase the music, view biographical information for the artist, etc.

Our application concept is analogous to this Pandora model. We enable users to create “Book Bags” to organize books they discover within the application in whatever manner they want. The user could create Book Bags such as “Books I Like,” “Novels for My Kids,” “Possible Christmas Gifts for Spouse,” “Romance Books I Want to Read,” etc. Then they can fetch recommendations for other books related to those books already inside a Book Bag. Our recommendation algorithm uses the Amazon Product API to find books relevant to those already in a Book Bag. These are cross-listed using the NYT API, so that the recommended books are presented in ranked order based on their success as NYT Bestsellers. Like Pandora, the user can select books they are not interested in, such that our algorithm gets "smarter" about the user's preferences the longer it is used. Moreover, the application provide the NYT review for all recommended books that have been reviewed by the NYT. Like Pandora, it also provides the link for the user to purchase the book on Amazon.com. Aside from Pandora, our product is also comparable to GoodReads. GoodReads is a great tool for organizing books of lists, but what makes our application special is the role of the NYT Bestsellers Lists and Reviews in providing quality assurance for curated lists of recommendations.

Once we had settled on this overall concept, we split up roles and responsibilities among our team. Lauren and Tom were assigned to the implementation of the application's functionality. Tom focused on integrating the NYT API with the Amazon Product API, and building the book recommendation algorithm. Lauren implemented the rest of the user interface, including the user profiles, Book Bags, and Bestsellers lists. Kristie spearheaded the process of iterative prototyping and she created mockups throughout the development process using LucidChart as an alternative to Balsamiq. Sarah was responsible for all of the documentation and writing for the final submission, and provided support to Kristie for prototyping. Everyone participated in the filming of our video, and Lauren did the video editing. We were all involved in the important design decisions, namely the "look and feel" of the application. Everyone contributed design ideas that are represented in our end product; no major features or changes were implemented without the approval of our entire team. We used Google Docs and Github to work collaboratively on design concepts and code, respectively. Throughout the process we stayed in contact via email and phone, and we met numerous times in person to ensure all team members were aware of everyone else's progress.

			
Tom Segarra	**Lauren Zou**	**Kristie Howard**	**Sarah Green**

TARGET USERS

Our targeted class of users is young adults (18-35). Since our application provides book recommendations based on user preferences and the New York Times Bestsellers' List, we are focusing on this demographic as the nexus of web app users and book readers. Many young people are tech-savvy, and would be likely to feel comfortable and enthusiastic using a web app like ours. We felt it was better to target young adults than teenagers, since reading for pleasure is a hobby that more adults have time to enjoy than students. In addition, young adults tend to rely heavily on expert and user-generated reviews in making decisions for cultural consumption (e.g. Netflix, Amazon Recommendations, Rotten Tomatoes), which our app provides. We think that young adults are the best-suited demographic for us to target, since they are among the most likely people to seek out curated recommendations for books, feel at ease with web apps, and actually have time to read for pleasure.

Finally, the "user" may comprise multiple individuals sharing one account, like couples or family members. To accommodate this possibility, the application "multi-profile" functionality. This way, individuals sharing an account may each create their own "profile" to keep their recommendations separate, while making it easy to switch between profiles. This feature is in accordance with industry standards and other similar applications, like Netflix, which allow multiple profiles to be shared by a single account.

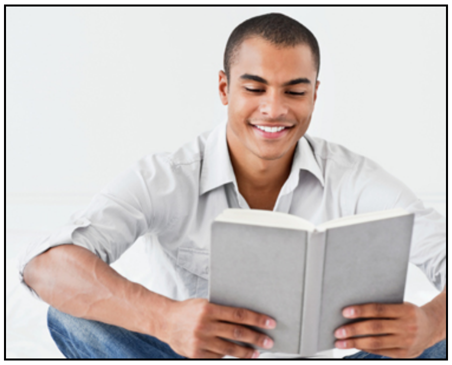
PERSONAS

BETHANY, 32 years old

Bethany is an avid book reader. She graduated as an English major from Wellesley College in Massachusetts and is now a stay-at-home mom with two toddlers. She always has her nose in a book. Recently, Bethany realized that she has exhausted all of her book lists, and she doesn't know what to read next. She has read all the books currently on the New York Times Fiction Bestsellers list, and wants to explore more books based on her preferences. She shares her Book Bags account with her husband; he and Bethany each have their own “profile” to keep their Book Bags and recommendations separate.

CASEY, 22 years old

Casey is a recent graduate of the University of Southern California with a degree in Business. He is a tech enthusiast and always has the latest iPhone. He has never had much free time to read for fun, but now that he’s out of school he wants to become more of a reader. He doesn't know where to start, but he is enthusiastic about using Book Bags to figure out what kinds of books he might like.



USE SCENARIOS

BETHANY

Bethany, already a user of the app, curates a list of books that she has read. Her husband, who shares her Book Bags account, edits the Book Bags saved to his profile.

STEPS

1. When Bethany opens the app, she sees a few “Book Bags” that she had created in a previous session. A Book Bag is a grouping of books. A book can belong to multiple Book Bags. Bethany's Book Bags are "Romance," "For Kids," and “Recent Reads.” In a previous session, Bethany had removed the "Books I Like" Book Bag (which is provided by default to the first-time user) and had added the custom Book Bags mentioned above.
2. Bethany clicks on her "For Kids" Book Bag and sees a list of all the books that she has added to that Book Bag. She also sees, below, a list of books that are recommended to her based on the books in this Book Bag. The recommended books that are New York Times Bestsellers are distinguished by a gold ribbon with “NYT” written across it, which is wrapped around the left corner of the book cover image.
3. Bethany sees a book she has not read before, and thinks that it would be a good book to read to her kids. The book is even a New York Times Bestseller! She adds the book to the “For Kids” Book Bag.
4. Bethany decides that she wants to find more books to buy for her kids. She uses the breadcrumbs to return to her "For Kids" Book Bag page.
5. Bethany exits the Book Bags application by closing the browser tab.
6. Bethany’s husband re-opens the Book Bags application, and switches to his profile. He clicks into his “Tech books” Book Bag. He renames the Book Bag “My tech books”, and deletes one book from the Book Bag.

CASEY

Casey opens the app for the first time, gives it information about his preferences, and chooses a book based on those preferences.

STEPS

1. Casey opens the app for the first time, hoping to find recommendations for books to read. He is prompted to enter his name and choose an avatar. Then he sees the welcome message and clicks to get started.
2. Casey sees on his dashboard a Book Bag called “Books I Like.” He clicks it, and sees that it is empty. He uses the back button to go back to the dashboard.
3. Casey looks at the tiles representing different NYT Bestseller Lists, and chooses the Business Books tile. This generates a list of the top Business books according to the New York Times Bestseller lists.
4. Casey clicks on a book cover, which causes a “More Info” panel to appear below the book. Casey closes the panel. Casey marks the book “Not interested” and sees it fade on the interface.
5. Casey sees a book that he has read before, and adds it to the Book Bag “Books I Like.”
6. Casey sees another Bestseller that looks interesting, clicks on “More Info”. Then he clicks on “NYT Review.” He reads the book’s New York Times review. He adds the book to a new Book Bag called “Books I Want to Read.”
7. Casey then decides he will buy the book, and clicks on the “View on Amazon” button.

DESIGN DECISIONS

Our design process was primarily influenced by the heuristics of minimalist aesthetics, flexibility and ease of use, user control and freedom, and recognition over recall. We aimed to create a product that had real-world utility, with a user interface reflecting this professionalism.

The overall page structure is essentially a persistent header, then the body that displays a variety of pages, and then a help link in the footer. We thought that the content in the header (logo, breadcrumbs, user profiles) was important enough that these be displayed persistently above page body no matter the user's scroll position. This way, the user can conveniently navigate, switch profiles, and know their location at any time without needing to scroll up. Our approach for placing content on the page was to exploit users' natural reading patterns, to ensure that they read as much content as possible when skimming the page. In our culture, this reading pattern is left-to-right and top-to-bottom, which was the flow we tried to imitate in our page content. We placed the header and breadcrumbs a bit further to the left, so the user would see these first. Then we aligned all the page content to the same left margin, so their eyes would not have to jump around as they progressed. Instead of using sidebars or another layout that involves placing unrelated content at the same horizontal level, we made sure to use horizontal sections that stretch across the page and expand downwards if needed. Not only is this in line with users' reading patterns, but it is also more friendly to mobile and tablet browsers, making our application more flexible to use across a variety of devices.

We chose our fonts early on, using these as a launching pad for other design decisions. The script font used widely across the application, Amatic, immediately jumped out at us. It seemed quirky and unique, evocative of books but not as clichéd for a book-related application as a font like Papyrus. We thought that choosing to use a font full of personality would promote recognition over recall for a returning user, since they would associate that font with our application. However, we thought that using this script font for all of the text in the entire application would be overwhelming to the eye and not minimalist. So we chose to use Open Sans, a much more generic and quiet font, to complement Amatic. We only used Open Sans for "system functions" such as breadcrumbs navigation, switching between user profiles, accessing the help documentation, as well as lengthier portions of text like book descriptions.

Next we decided on the overall “look and feel” of the application. We opted for a “flat design,” whereby there are no box shadows or other design aspects causing elements to jump off the page. We noted that this “flatness” is a minimalist and very trendy aesthetic. We were influenced in this regard by Apple, who have recently made this shift towards flatness with the new Yosemite interface for Macs. Another source of influence for our flat design was the Windows 8 user interface, which heavily features flat “tiles,” and uses sharp corners rather than rounded edges. We implemented this idea of tiles for our Book Bags and Bestsellers blocks on the user’s dashboard. For Book Bags, we customized the tiles to literally look like bags, a design choice that makes the functionality of these elements clearer. We chose to borrow the Windows 8 usage of sharp corners as well, to achieve a cleaner feel throughout. We were also influenced by Google Material Design, which is a design language released by Google earlier this year. It is based on “paper and ink” materials, emphasizing the use of grid-based layouts, padding, and depth effects like lighting and shadows. Google Material Design influenced us to use handmade grid layouts, and to play with highlights and shadows in order to mimic depth without sacrificing “flatness.”

Our color scheme was chosen in the same spirit as the Amatic script font, promoting recognition over recall of our application by featuring fun and memorable combinations of colors. We also took care to choose colors that would make it easier for a visually impaired or colorblind user to distinguish areas of the user interface. All the colors we used were pulled from http://flatuicolors.com, which is a site that provides a small set of colors known to complement one another and be aesthetically pleasing to the eye. We opted for a dark purple background to offset the bright colors of the buttons and other page elements, while remaining more interesting and original than a grey or navy background. We used white as the text color consistently throughout the application, since it stands out against all the bright colors used elsewhere.

We made several smaller-scale design decisions that we feel contribute to the professionalism of our application. Firstly, we included a handmade progress bar that appears immediately below the header. It unobtrusively promotes the visibility of system status as pages load. We included two ways to go back to a previous page: clicking the browser’s back button, or navigating using our handmade breadcrumbs in the header. Providing these two options for backwards navigation increases the application’s flexibility. The breadcrumbs also make the “emergency exit” clear for users who accidentally navigated to a page they didn't intend, which is in line with the heuristic of user control and freedom. Another way we supported "undo" was by making it possible for a user to mark a book as "Not Interested" (causing it to fade into the background) and then undo this action by marking the book as "Interested" if they change their mind. The user can also delete a Book Bag, if they mistakenly create one and wish to undo that action. However, the user is always prompted to confirm the deletion of a Book Bag, to address the heuristic of error prevention. The user can also undo adding a book to a Book Bag by clicking "Add to Book Bag" and uncheck the Book Bag in question.

We also designed our own browser favicon, based on the icon for a Book Bag. The favicon promotes recognition over recall for users with multiple browser tabs open, so they can easily locate and return to our application by recognizing the Book Bags icon. In addition, we implemented cursors with extra care, to ensure that clickable or editable content is distinguished from non-interactive content as the user moves their mouse around the interface. Additionally, we made sure to include, on hover, a slight change in color for buttons, drop-down menus, and other clickable areas. This helps the user easily learn how to interact with the interface and quickly become an expert. Finally, we took the overall approach of minimizing text as much as possible, such that there is no extraneous or overly verbose information in the interface. When text was necessary, we used a casual and welcoming tone; we thought the user would be more likely to trust the application's book recommendations if its general tone was friendly.

The aforementioned design decisions comprise the final "look and feel" of our Book Bags application. We ended up scrapping several of our earlier ideas, as we did our iterative prototyping and usability tests. For example, we originally thought to get a "More Info" box to appear on hovering over a book cover, similarly to the functionality of the Netflix interface. However, we realized that this type of pop-up would block a good deal of content behind it, and be less user friendly. We also anticipated, at first, implementing a flagging functionality for users to "mark as read" books they had already finished. While we thought that this flagging functionality might occasionally be useful, we also realized it was extraneous and compromised the application's simplicity. Moreover, we could not decide how the "mark as read" information would contribute to our book recommendation algorithm in a clear and meaningful way. Although we ended up scrapping these concepts (among others) from our final product, they were crucial to our design process.

PROTOTYPING & TESTING PROCESS

We began our prototyping process using pen and paper during our brainstorming sessions, to be able to iterate rapidly with little effort. Once our initial design concept was in place, we began using LucidChart to create hi-fi prototypes. We chose to use LucidChart over Balsamiq, because LucidChart allows collaborators to see each others' changes to a document in real time, like Google Docs. For each prototype iteration we first discussed what features needed to be mocked up on each page, then we observed and analyzed how other web applications have designed this type of feature, and then we put together the mockup itself. We made the mockups interactive, so that we could simulate the user experience of the application without writing any code.
Our LucidChart mockups can be found at:
https://www.lucidchart.com/documents/view/fc9b8ef3-477d-4c62-afb4-16c1db9ad9c7 (Version 1)
https://www.lucidchart.com/documents/view/7ba50689-9d29-4352-9a57-51c89db8ba0e (Version 2)

We performed user testing throughout our process. At first we took turns acting as the user, while the application was still heavily in development. We thought that an external, first-time user would be too blocked and distracted by the unfinished portions of the user interface to be able to give us much helpful feedback. Once the skeleton of the application was more fully implemented, we began doing user studies with both fellow computer science students and also non-tech savvy individuals. We conducted our studies by letting the tester interact freely with the application, including the first-time-user experience of creating a profile. We observed and took notes, providing clarifying answers to questions when necessary. We also administered a usability survey after the test was completed. The survey we used is pasted after this section in this document, followed by two completed surveys from our test users.

Below are some of our notes from our two most helpful usability studies:

Observations from user study with Huey:

- Didn't understand logic behind the icons
- First spent time looking at Bestsellers
- Asked "What do I do?" when looking at Bestsellers
- Clicked a book cover thinking would bring up information

- Expects to close More Info panel by clicking “More Info” again
- Wants access to ratings for books
- Breadcrumb navigation was well-understood

Observations from user study with Aftab:

- Felt overwhelmed about what to do after entering name/avatar
- Purpose of application was not immediately clear
- Found it visually interesting--fonts, colors--but thought these obscured the application’s functionality at first
- Did not make any errors
- Confused about why reviews appear for some books but not others

After the study with Huey, we opted to change how the “More Info” panel worked. We changed it such that there were two points of entry, rather than one. The user could click the “More Info” button, *or* click on a book cover, to bring up the panel. We also made it possible to close this panel by clicking anywhere on the screen. We began thinking about adding more information for the first-time user about how to get started.

After the study with Aftab, we decided definitively to create a better experience for the first-time user. While he found the application visually appealing, he felt that it was too explosive for the first-time user without instructions to get started. For this reason, we added a welcome message after the user enters their name and chooses an avatar, which explains the basics of our application and how to begin using it.

USABILITY SURVEY

How much do you agree with the following statements?

I always knew how to do what I wanted.

1 2 3 4 5 6 7 8 9

Strongly disagree Disagree Neutral Agree Strongly Agree

The app felt trustworthy.

1 2 3 4 5 6 7 8 9

Strongly disagree Disagree Neutral Agree Strongly Agree

The app is visually attractive.

1 2 3 4 5 6 7 8 9

Strongly disagree Disagree Neutral Agree Strongly Agree

I felt like I could do what I wanted.

1 2 3 4 5 6 7 8 9

Strongly disagree Disagree Neutral Agree Strongly Agree

The guided tour was helpful.

1 2 3 4 5 6 7 8 9

Strongly disagree Disagree Neutral Agree Strongly Agree

I felt like I could control the app.

1 2 3 4 5 6 7 8 9

Strongly disagree Disagree Neutral Agree Strongly Agree

I knew when I made errors, and how to fix them.

1 2 3 4 5 6 7 8 9

Strongly disagree Disagree Neutral Agree Strongly Agree

The app used language I felt comfortable with.

1 2 3 4 5 6 7 8 9

Strongly disagree Disagree Neutral Agree Strongly Agree

The app looked and felt like other apps I use.

1 2 3 4 5 6 7 8 9

Strongly disagree Disagree Neutral Agree Strongly Agree

Please write any additional questions or comments.

USABILITY SURVEY (Huey)

How much do you agree with the following statements?

I always knew how to do what I wanted.

1 2 3 4 5 6 7 [✓ 8] 9

Strongly disagree Disagree Neutral Agree Strongly Agree

The app felt trustworthy.

1 2 3 4 5 6 [✓ 7] 8 9

Strongly disagree Disagree Neutral Agree Strongly Agree

The app is visually attractive.

1 2 3 4 5 6 7 [✓ 8] 9

Strongly disagree Disagree Neutral Agree Strongly Agree

I felt like I could do what I wanted.

1 2 3 4 5 6 [✓ 7] 8 9

Strongly disagree Disagree Neutral Agree Strongly Agree

I felt like I could control the app.

1 2 3 4 5 6 7 [✓ 8] 9

Strongly disagree Disagree Neutral Agree Strongly Agree

I knew when I made errors, and how to fix them.

1 2 3 4 5 [✓ 6] 7 8 9

Strongly disagree Disagree Neutral Agree Strongly Agree

The app used language I felt comfortable with.

1 2 3 4 5 6 7 [✓ 8] 9

Strongly disagree Disagree Neutral Agree Strongly Agree

The app looked and felt like other apps I use.

1 2 3 4 5 6 [✓ 7] 8 9

Strongly disagree Disagree Neutral Agree Strongly Agree

Please write any additional questions or comments.

USABILITY SURVEY (Aftab)

How much do you agree with the following statements?

I always knew how to do what I wanted.

1 2 3 4 5 6 [✓ **7**] 8 9

Strongly disagree Disagree Neutral Agree Strongly Agree

The app felt trustworthy.

1 2 3 4 5 6 7 [✓ **8**] 9

Strongly disagree Disagree Neutral Agree Strongly Agree

The app is visually attractive.

1 2 3 4 5 6 7 8 [✓ **9**]

Strongly disagree Disagree Neutral Agree Strongly Agree

I felt like I could do what I wanted.

1 2 3 4 5 6 [✓ **7**] 8 9

Strongly disagree Disagree Neutral Agree Strongly Agree

I felt like I could control the app.

1 2 3 4 5 6 7 [✓ **8**] 9

Strongly disagree Disagree Neutral Agree Strongly Agree

I knew when I made errors, and how to fix them.

1 2 3 4 5 6 7 [✓ **8**] 9

Strongly disagree Disagree Neutral Agree Strongly Agree

The app used language I felt comfortable with.

1 2 3 4 5 6 7 [✓ **8**] 9

Strongly disagree Disagree Neutral Agree Strongly Agree

The app looked and felt like other apps I use.

1 2 3 4 5 [✓ **6**] 7 8 9

Strongly disagree Disagree Neutral Agree Strongly Agree

Please write any additional questions or comments.

Bright colors, font, and tiles -- visually interesting but not immediately understandable. Couldn't figure out for a while what the app does.

SOFTWARE ENGINEERING

Our Book Bags application was built using HTML5, CSS3, and JavaScript, with the addition of several external libraries and tools. These third-party resources were used to assist our workflow, simplify and beautify our code, facilitate compatibility across browsers, store data across user sessions, and generally make the implementation of our product design feasible. Per the assignment guidelines, we began by choosing to work with one of the New York Times APIs, the Books API. We used both the Best Sellers and Reviews request types, to generate lists of bestselling books and their corresponding NYT reviews, respectively.

We used Node.js and a couple of node modules in order to make our development process easier. We used the Less module, which makes collaborative development easier by allowing for variables and mixins in our CSS. Less facilitates consistency with regard to styling during collaboration. For example, variables can be used to store a hex value, so the same shade of a color can used consistently across all stylesheets by referencing that variable. We also used the Autoprefixer module for cross-browser compatibility, to free us from manually writing in the prefixes for CSS properties to accommodate any browser. For instance, instead of writing "display: -webkit-box; display: -moz-box; display: -ms-flexbox; display: -webkit-flex; display: flex;" for the flex-boxes in our user interface, Autoprefixer let us simply use "display: flex;". Node.js was used to compile our Less code, orderly split into multiple files for ease of development, into a single CSS stylesheet. Then Node.js could prefix the CSS code using Autoprefixer. Specifically, we wrote a compiling tool called Compile.js that wraps around the Less compiler and Autoprefixer compiler. Within this tool, we used the module node-watch to listen for developer changes to the Less files, which trigger automatic compilation into CSS for a seamless workflow. We also used the module "q" for using deferred objects. Finally Node.js and Autoprefixer were used to minify our CSS code, which allows our pages to load faster and provide a better user experience.

We used four primary libraries to supplement our application's "plain" JavaScript. First, we used jQuery widely across our scripts for the sake of brevity, flexibility, and professionalism. We also utilized the jQuery UI library to assist with animations and transitions. For example, we used functions from this library to create a smooth transition in the interface between the Dashboard page and a NYT Bestsellers list page.

Another library we used is the Store.js JavaScript library, to store data in the browser across user's sessions. Store.js enables the app to store data about the current user, including their name, their Book Bags, and the list of Books they have marked as "Not Interested." This usage of Store.js both addresses the assignment guidelines and also ensures that returning users do not have to start over from the beginning. Moreover, storing data permits our algorithm for recommending books to improve over time as it gathers more data about the user's book preferences. Finally, we used the Clamp.js library to format multi-line text that overflows from a box. This came into play for displaying book titles and authors. Clamp.js adds an ellipsis to hide the overflow and indicate there is content beyond the ellipsis being truncated.

The last third-party component we used was the Amazon Product Advertising API, to expand on the functionality of the NYT Books API. After the NYT API is used to fetch NYT bestselling books in a variety of categories, the Amazon API is called in order to fetch similar books to recommend to the user alongside the bestsellers. We had originally planned to use the GoodReads API to generate these lists of related books, but found that the GoodReads API requires 1 second to pass between each call. We opted to use the Amazon API instead, since it provides the same functionality but is much more liberal in terms of calls. To use the Amazon API we also had to use a JavaScript library called sha2 to "sign" each requests to the Amazon API. We also used the JavaScript library Moment.js to format the timestamp of each request to Amazon API. Usage of these two small libraries ensures that each request to the Amazon Product Advertising API is secure, per that API's documentation.

On the following page is a tree mapping the structure of our code repository, in order to make it easier to locate any particular file(s) for grading. Our code can also be found online at https://github.com/kbh2120/BookBags in a private repository.

CODE REPOSITORY STRUCTURE

```
BookBags
        amazon-test
                index.html
                scripts.js
        assets
                css
                        font-awesome.min.css
                        jquery-ui.min.css
                        styles.css
                fonts
                        FontAwesome.otf
                        fontawesome-webfont.eot
                        fontawesome-webfont.svg
                        fontawesome-webfont.ttf
                        fontawesome-webfont.woff
                images
                        avatar-1.png
                        avatar-2.png
                        avatar-3.png
                        avatar-4.png
                        blank-book-cover.png
                js
                        amazon.js
                        api.js
                        bestseller-list.js
                        book-bag.js
                        book-review.js
                        book.js
                        cache.js
                        clamp.min.js
                        jquery-2.1.1.min.js
                        jquery-ui.min.js
                        moment.min.js
                        pages.js
                        progress-bar.js
                        recommender.js
                        scripts.js
                        sha2.js
                        store.min.js
                        user.js
                less
                        account-creation.less
                        bestseller-lists.less
                        book-bags.less
                        book.less
                        help.less
                        main.less
                        styles.less
        README.md
        compile.js
        favicon.png
        index.html
```