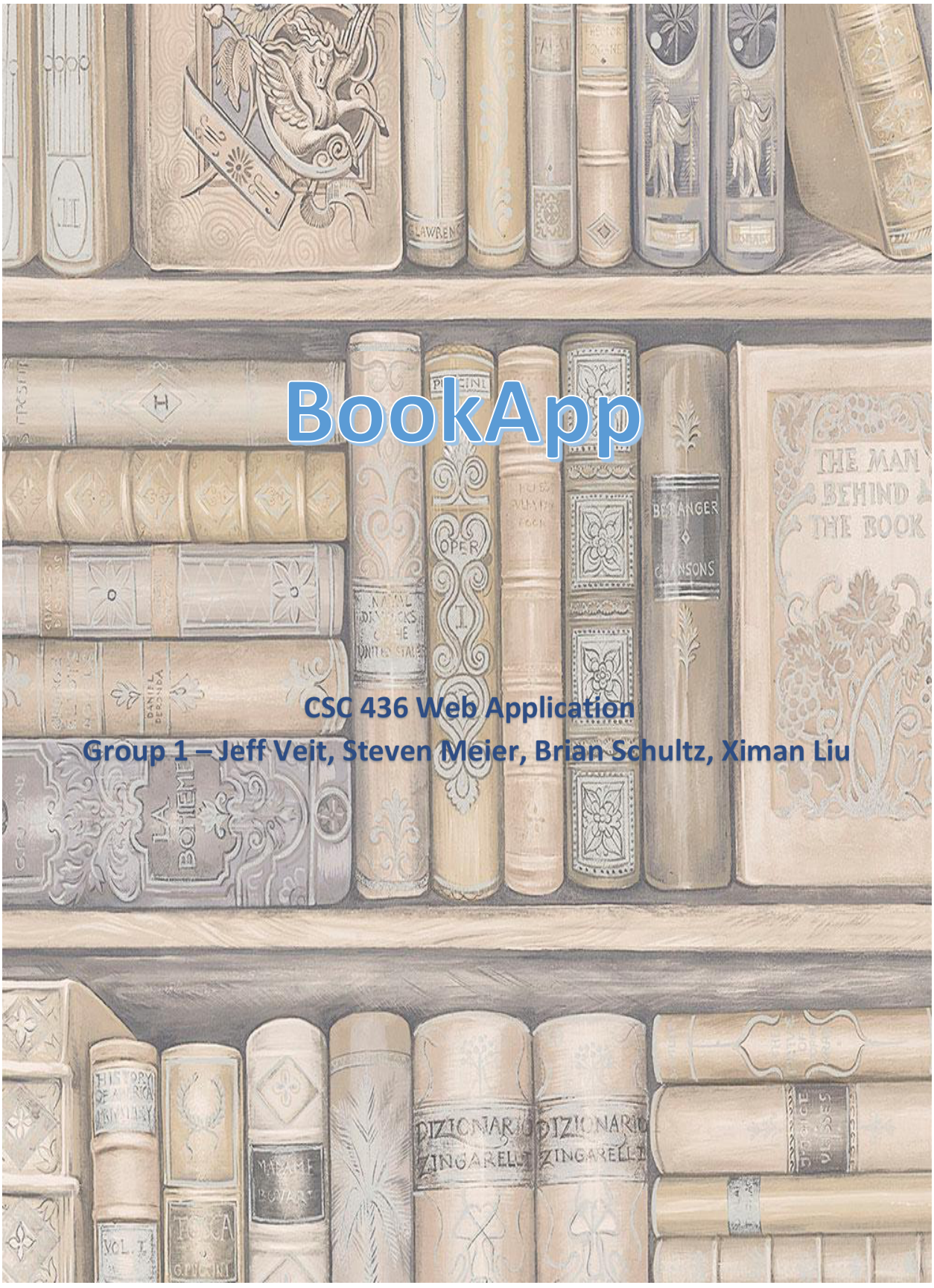
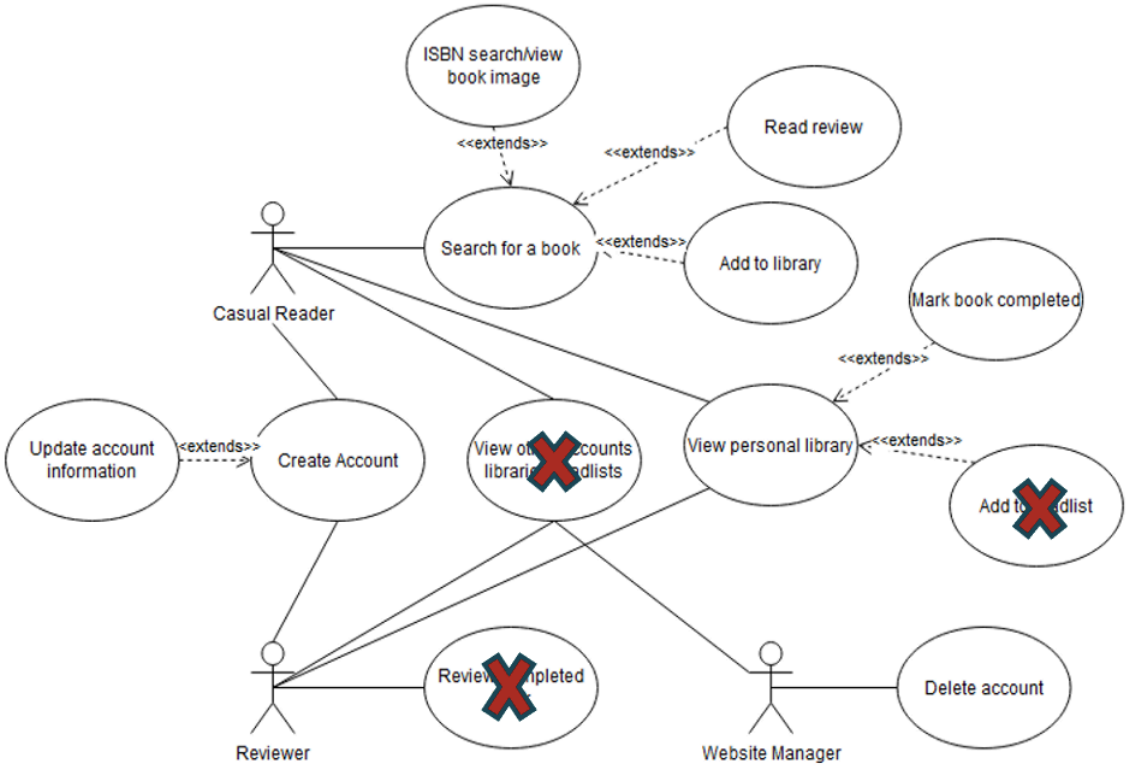

BookApp
CSC 436 Web Application
Group 1 – Jeff Veit, Steven Meier, Brian Schultz, Ximan Liu

Milestone 1

Overview:

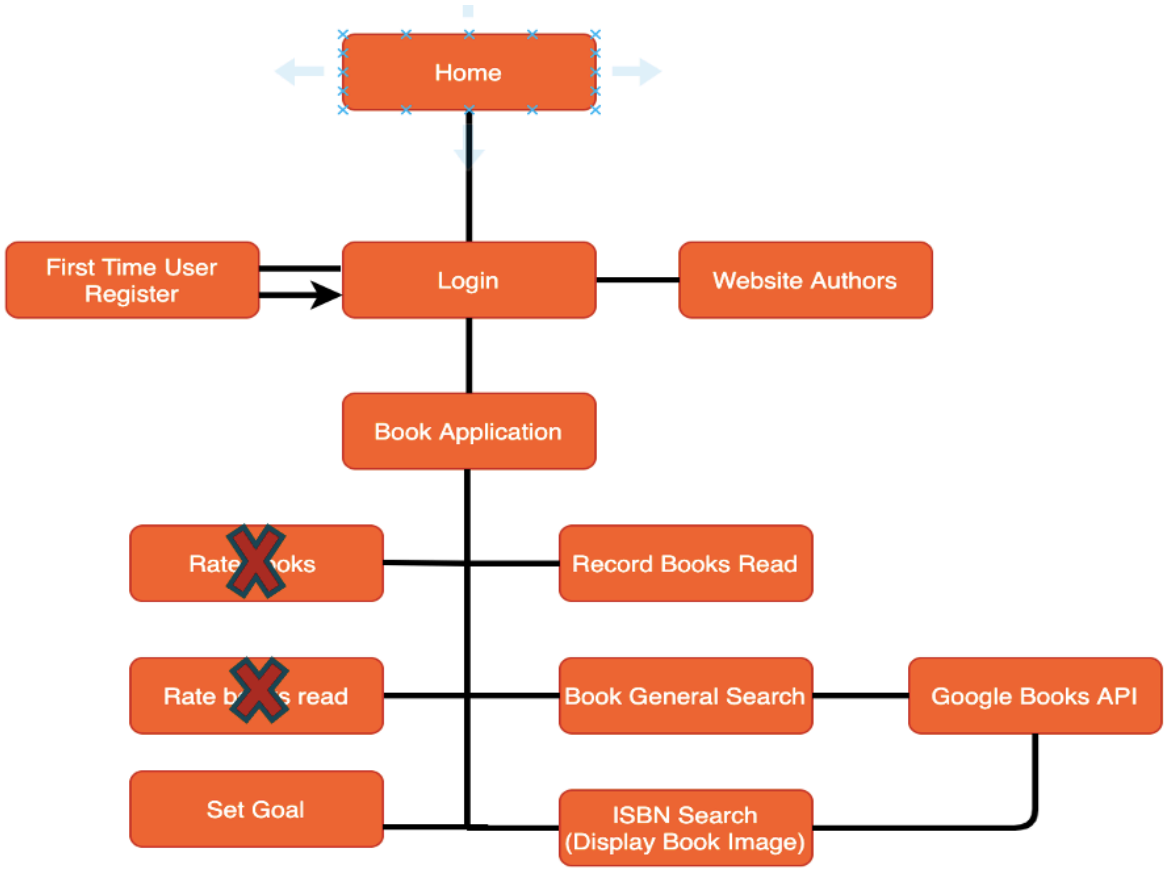
The project being created will allow users to search and find books through the web application, while also tracking reading goals and recording what has been read. Casual readers have the ability to do a general search by category or key word and get a bunch of responses based on their entry to look at. If they would like to search by a specific ISBN, they can look up a book that way and view an image of it. Some of the more enthusiastic readers like to provide reviews and track what books they have read so they can meet their personal goal. Users have their own accounts that they must create so their specific goals/books can be tracked.

Use Case Diagram:



User Navigation:

When users first hit the home page, they will see the authors of the project and have the ability to login or create a profile. Once logged in, they have the ability to do book searches by general category or ISBN, set goals, record books read, or rate the books they have read. Both search functions will use the Google Books API.



Persona Definitions:

Persona 1
Name: Bobby Binary
Age: 27
Location: Arlington, Virginia
Education: Bachelors in Network Engineering
Job: Systems Administration
Family: Single, parents live in Minnesota

Goals for Reading:

(1) Finding new books in the technology-instruction genre that are worthy of his time and money

(2) Discovering new novels in the techno-thriller and political-thriller genres

Frustrations with Reading:

(1) Keeping track of his books and bookmarks in the technology-instruction genre, as he would like to use for reference while at work

(2) Difficulty in tracking reading goals

Reading Habits:

(1) Prefers technology-instruction books that are free
(2) likes to review books to share thoughts with other readers
(3) reads exclusively eBooks on his work computer, home computer, and mobile phone
(4) listens to audiobooks in the techno-thriller and political-thriller genres during his commute
(6) Never re-reads a novel, but does enjoy books in a series, such as Tom Clancy's books centered on the Jack Ryan character
(5) Sets yearly goals for book reading and posts the results on social media

Persona 2
Name: Jane Sequel
Age: 44
Location: San Diego, CA
Education: Master's in English
Occupation: English Teacher
Family: Married with kids

Jane is an English teacher at a local high school and enjoys reading all kinds of literature. She loves to bring that enthusiasm for reading to her students and fills them in on new books she has read. She uses the web application to search for books by a general subject, hoping to find something that catches her eyes.

She also likes to keep track of what she has read. Since she is a motivated reader, she likes to set a goal to make sure she reaches it. She considers herself a good judge of a book given her credentials, and thoroughly enjoys providing reviews on everything she has read. By doing this, she hopes that others will dive in on those literary treasures she also loved.

External Service:

Google Books API
Website: https://developers.google.com/books/docs/overview
Data Format: JSON
Calling Style: REST
Example Call: GET https://www.googleapis.com/books/v1/volumes?q=quilting

The Google Books API v1 gives you programmatic access to many of the operations available on Google Books website. You can use it to create powerful applications that provide deeper integration with Google Books. Some of the main features that the API provides are:

- Search and browse through the list of books that match a given query.
- View information about a book, including metadata, availability and price, links to the preview page.

- Manage your own bookshelves.

Books API operations

You can invoke five different methods on collections and resources in the Books API, as described in the following table.

Operation	Description	REST HTTP mappings
list	Lists a specified subset of resources within a collection.	**GET** on a collection URI.
insert	Inserts a new resource into a collection (creating a new resource).	**POST** on a collection URI, where you pass in data for a new resource.
get	Gets a specific resource.	**GET** on resource URI.
update	Updates a specific resource.	**PUT** on resource URI, where you pass in data for the updated resource.
delete	Deletes a specific resource.	**DELETE** on resource URI, where you pass in data for the resource to be deleted.

Milestone 2:

Feedback Incorporated:

Person providing Feedback	Description	Incorporation
Ken Yu	"For next milestone, you may need to narrow or refine the scope and so as part of the documentation, make sure the team discuss why you picked one and not the others so you can look back at the end of the quarter for the implication of the decision."	For Milestone 2, we decided to refine the scope by cutting back on some of our use cases described in Milestone 1. The personal library linked to a user's google account was removed, along with writing reviews and viewing other users book libraries. We removed these elements because they would take considerable work to implement. The main use cases of the application such as book search, goal tracking, and account registration all remain. We felt those were the main focus of the site.

Milestone 3

Link to GitHub: https://github.com/jav-1186/bookApp

Link to GitHub Pages: https://jav-1186.github.io/bookApp/home

Changes:

Change	Old Solution	New Solution
Login Page	Had 3 potential login pages that would then route to the application.	Now just having login functionality directly on the homepage. When we incorporate the data source into the project a solution will be created where personal information pages show default or no information until logged into an account.
Header formatting	Nav bar HTML written out on every reachable page (excluding the login pages).	Now using the Header component to manage all our navbar/logo logic and formatting.
Footer formatting	Footer information written out on every reachable page (excluding the login pages).	Now using the Footer component to manage all our footer logic and formatting.
Index.html	Essentially the home page of the web app. After passing through a login screen you reach this page where you can then use the navbar to link to different pages	Sets header information for browser tab display. Routes directly to app-root (app.component.html).

App routing	Was managed through the copied and pasted header navbar on every reachable page.	Gives us the ability to represent any component we wish to create or allow navigation to. The header component creates nav links that are then tied to the routes defined in the application routing. Represented in the app component section as <router-outlet>.
App component	Did not exist.	Is the app-root that our index.html displays. Formats the skeleton of the site, putting the header above the router-outlet functionality, followed by the footer formatting.
Main component	Was managed through main.html.	Now is its own module with typescript, css, and html file. Still displays the same information as the old main page, but now, with type-script, has the ability to integrate forms for data submission once we integrate with our data/api service.

Library component	Was managed through personalLibrary.html.	Now is its own module with typescript, css, and html file. Still displays the same information as the old library page but with typescript will now be able to pull personal account information and use forms to create new library entries.
Goals component	Was managed through goals.html.	Now is its own module with typescript, css, and html file. Still displays the same information as the old goals page but with typescript will now be able to pull goals for your account and use forms to create new account goals.
Home component	Was managed through index.html before.	Now is own module completely separated from the index.html file. With the typescript this page will be responsible for submitting login information to the service. As well as displays information bout the creators of the application.

Account component	Was managed through the account.html page before.	Displays the account information page just as before. Not much will change with this component, but typescript will be used to display stored account information.
Angular Integration	None before.	By creating this as an angular framework project we can now use commands like “ng serve” to boot up our websites locally. Using angular also gives us a better folder structures so we now have more separation between functional parts and application assets.

Milestone 4

GitHub Source Code: https://github.com/jav-1186/bookApp

GitHub Page: https://jav-1186.github.io/bookApp/home

Overview and Changes:

The web application has now integrated the Google Books API service and has three different API calls. A web service component was created and handles all of the API calls. We had to get a Google API Key and register it with the Book API. This is used to authenticate with the server and is appended to the URL.

The three API calls are for:

- ISBN
- Book Title
- Book Subject

We also were able to get Firebase integrated and some basic authentication in place. A user can login with google credentials and utilize the application.

Within the structure of the code, we separated the search and login functionality into their own components. This made the code cleaner and easier to manipulate those specific functions. We also changed how bootstrap and jQuery being used within the application. Previously we were calling them from the application through links in the index.html file, but we have now installed the tools locally. This was done because it fixed certain bootstrap problems we were having.

From here we need to finish the firebase implementation and fine tune some of the functionality. The majority of it is in place and we feel good about completing everything on time.

Lessons learned

- Have consistent capitalization/naming conventions right at the start
    - Created issues between machines being ok with pascal/camel mix and then some machines refusing to function when import statements didn't follow same structure

- Allow for extra development research time when integrating a 3rd party tool
    - Had issues getting main.yml working with the github workflow and firebase took quite a bit of troubleshooting to get it worked out
    - Had issues getting user information out of auth service for use when creating/editing user collections in firestore
    - Documentation in both above cases was not all that helpful and solutions for our problems found on stack overflow bared little resemblance to documentation suggestions

- Ensure you are using up to date versions
    - Github workflow stopped functioning due to a depreciated command. Took quite a bit of troubleshooting to correct.

- Be wary of how ts API calls work asynchronously
    - Our function completes its work calling the API and doesn't receive an error, so it continues working through and completing without waiting for the API call to complete
        - This messes up session storage read/writes and creates race conditions
    - Need to become more comfortable using promises and async in ts

What you would do differently

- A clearer layout of the website to begin with
    - kind of inconsistent in how components are made and the usability is a bit jumbled on the site
    - Lacks clarity in what components are responsible for and minimizing the amount of navigation
    - have more of a plan could have allowed better division of work amongst group members

UX Design

- Our initial design I think was functional/reliable but at this point in the development I think we are solidly in reliable approaching usable

- We could continue improving our standing by eliminating/combining some of the smaller components into singular pages
    - Could probably condense the site down to 3 or 4 navigational routes