Bookshelf.init
Software Development Group Project

Team Members: Aidan Reese, Alex Barry, Brody Cyphers, Tyler Coil, Luca Hodge

Project Description:

Bookshelf.init is a website designed for students, where users can upload study guides, favorite documents that they find helpful, and search for resources in a specific subject.

Users are able to register an account on the website, which creates a profile page that displays uploaded material as well as records of the documents that the user has liked.

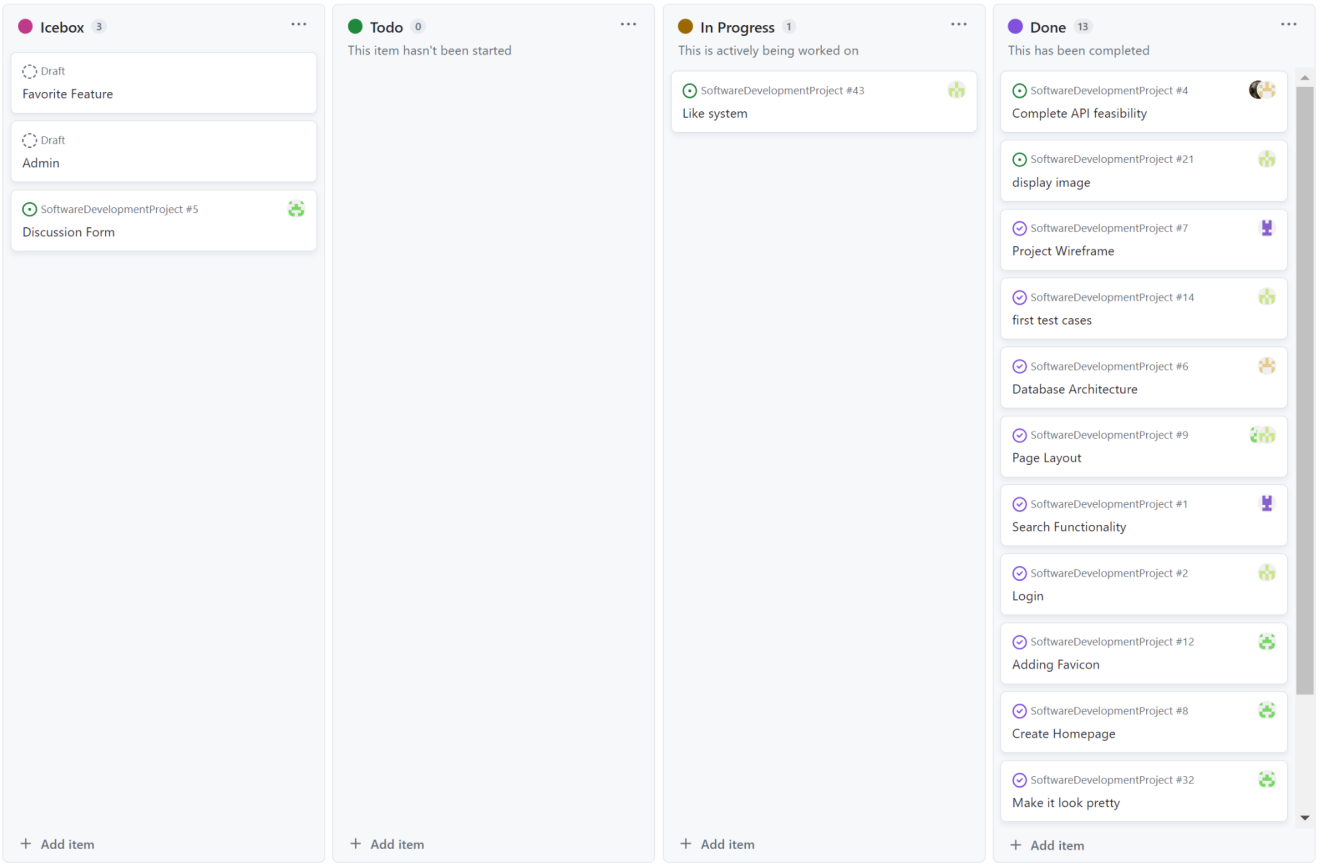
The upload page allows users to select any file type of their choosing, and associate it with a subject and tags in order to categorize their study guide. Once the user submits the form, their study guide is uploaded to the website and the document will appear in their profile page.

The website also allows for users to search for uploaded study guides through criteria like name, tags, or by subject. Depending on the user's input, the page returns all relevant uploads, where the user can then view the files and favorite them in order to save them to their profile.

Together, these features can be used to store, organize, and discover useful study sources. This project serves as a collaborative platform to assist any users in their studies.

Link to Project Tracker: https://github.com/users/lucakhodge/projects/1

GitHub Project Board:



Project Video: https://youtu.be/bCL_Ef1a8qA

Version Control System: https://github.com/lucakhodge/SoftwareDevelopmentProject

Contributions:

Aidan: Upload functionality and API that saved different types of tags, subjects and the name of the file that was being uploaded. Implemented a functioning file storage API, Upload Care, that allowed us to deliver files as a link that had been uploaded by the user to our other pages. Also designed the whole postgres database to store Users, Study Guide information, Tags and Subjects. As Well as reference tables that connect Subjects/Tags to given Guides.

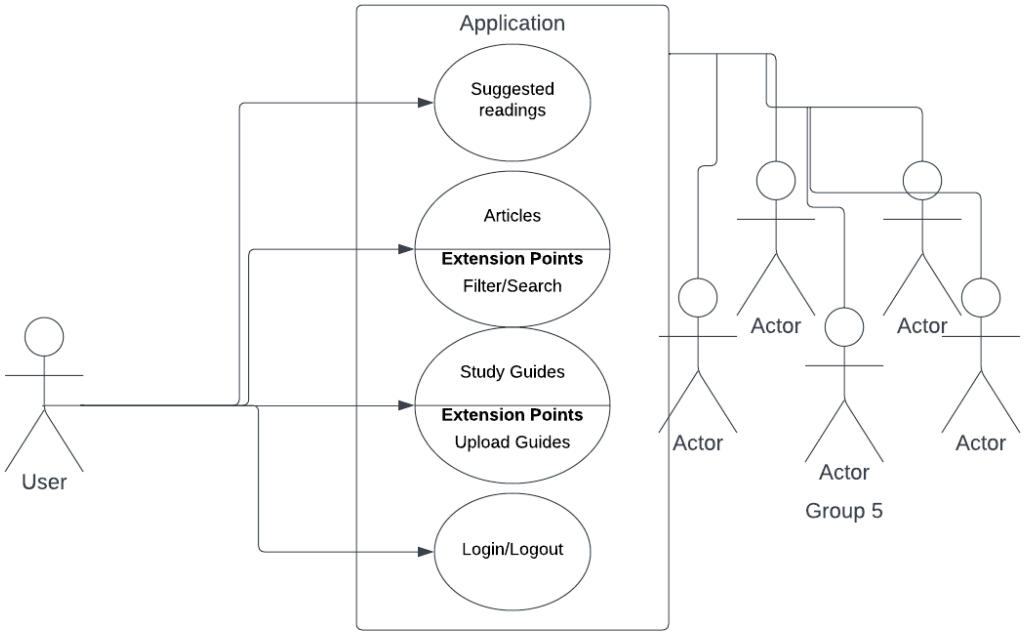
Alex: Upload functionality, Storage API, File encoding. The API was the most time consuming, as there were issues with pricing and storage limits. There was also work done to make the pdf uploads converted to a different format (BLOB) in order to insert the file into the database so that it could be saved externally and properly pulled. The upload functionality was done using express, so that the file information could be extracted as well.

Brody: Created the search feature, which involved designing an HTML get request form which sent user input to an custom API route. The API then joined all relevant tables and built an SQL query based on the user input, which could accept any combination of filters, tags, or keywords. Information returned from the database was then parsed and displayed on the results page using EJS functions. Search results linked to the corresponding display page, which then updated database values if the document was liked. Also designed website wireframes and architecture diagrams, ran user and integration testing, and contributed to site design.

Tyler: Contributed to creating the Home page and Discovery page which involved using API's, HTML, CSS , and NodeJS. In addition, I made it possible for the site to use static files which allowed images, CSS files, and favicons. Designed the navigation bar, footer, favicon, and Logo. Updated the Login, Register, Search, and Upload pages to add a more esthetic look for the site

Luca: Register, Login, Logout, Profile page, Display page, Liking system, Test cases. The register, login, and logout features involved setting up a small portion of the database and adding user info to it. The profile page involved ejs, javascript, and queries to the database. The display page used an iframe to display the documents and images. The liking systems also queried and updated the database. Finally, the unit test cases used mocha and chai.

Use Case Diagram:



Test Results:

Unit Testing

```
Server is listening on port 3000


  Server!
Database connection successful
    ✓ Returns the default welcome message (95ms)
(node:54) [DEP0066] DeprecationWarning: OutgoingMessage.prototype._headers

(Use `node --trace-deprecation ...` to show where the warning was created)
    ✓ positive : /register (796ms)
2023-05-02 22:11:27.344 UTC [73] ERROR:  duplicate key value violates uniq

2023-05-02 22:11:27.344 UTC [73] DETAIL:  Key (username)=(abc) already exi

2023-05-02 22:11:27.344 UTC [73] STATEMENT:  INSERT INTO users VALUES ('ab
lp1unw417kg//O/Sf7SrFVH0sN9V6gLhiEq') returning *;
    ✓ negative : /register (450ms)
Error [ERR_HTTP_HEADERS_SENT]: Cannot set headers after they are sent to t

    at new NodeError (node:internal/errors:400:5)
    at ServerResponse.setHeader (node:_http_outgoing:663:11)
    at ServerResponse.header (/home/node/app/node_modules/express/lib/resp

    at ServerResponse.send (/home/node/app/node_modules/express/lib/respon

    at done (/home/node/app/node_modules/express/lib/response.js:1035:10)
    at tryHandleCache (/home/node/app/node_modules/ejs/lib/ejs.js:280:5)
    at exports.renderFile [as engine] (/home/node/app/node_modules/ejs/lib

    at View.render (/home/node/app/node_modules/express/lib/view.js:135:8)
    at tryRender (/home/node/app/node_modules/express/lib/application.js:6

    at Function.render (/home/node/app/node_modules/express/lib/applicatio

    at ServerResponse.render (/home/node/app/node_modules/express/lib/resp

    at /home/node/app/index.js:253:8
    at process.processTicksAndRejections (node:internal/process/task_queue

    ✓ positive : /login (716ms)
    ✓ Negative : /login. Checking invalid name (182ms)


  5 passing (3s)
```

Testing Login Functionality: Positive and Negative Cases
Testing Register Functionality: Positive and Negative Cases
Testing Repeated Registration:Negative Case

User Acceptance Testing

Use Case 1: Login and Register

As an end user, I should be able to register an account with unique credentials and login using this information to interact with the website.

All test users during this phase were familiar with the process of registering and logging in, and their actions matched our expectations. Users are automatically sent to the login page when clicking “Get Started”, and most successfully navigated to the register page if they knew they didn’t already have an account. If they tried to register while still on the login page, an error message notified them and they corrected themselves. Users were also able to realize when their credentials were not unique when registering. No changes were made to this functionality.

Use Case 2: Upload Documents

As an end user, I should be able to upload a document to the website and have it saved in the database for other users to view.

When navigating to the upload page, many users defaulted to first adding a file from their local computer and then filling out the file information. This was interesting, as the “Add File” option was not the first on the page and the visual weight of each field was the same. When asked, users reported that they attached a file first because it was the primary function of the page and they skimmed over other fields to find the upload option. Some attempted to submit without filling out any additional information, which prompted us to mark the other fields as required.

Use Case 3: Display Profile

As an end user, I should be able to view my profile page when logged in to see my information and badges along with liked and uploaded documents.

There was initially some user confusion on where to find saved and uploaded documents, as users did not associate their profile with documents, but rather with personal information as is common with other websites. To remedy this problem, we chose to redirect the user on a successful login to their profile page, which contained fields informing them of the purpose of the page and documents that could be stored there. Once this change was implemented, their behavior was consistent with the use case.

Use Case 4: Search Documents

As an end user, I should be able to search through all user uploaded documents to find relevant results, and then be able to view and save a document.

Testing this use case revealed many unexpected behaviors, as all users had a different approach to searching. Some wished to view all documents hosted on the website, while others attempted to narrow down the results as much as possible. The reasoning behind these decisions was reduced to personal preference, and the search page had to be changed to allow all possible actions. This process was repeated several times until all users were able to search how they wanted and were satisfied with the results.

Deployment: http://recitation-012-team.eastus.cloudapp.azure.com:3000/search