

Team CSJ

Elevator Pitch

Textbook costs impose many burdens on college students. The average college student expends between \$628 and \$1,471 annually on textbooks and other supplies ([Education Data Initiative, 2022](#)). Specifically, GW students are expected to spend around \$1,400 annually on books and supplies. According to a survey of more than 5,000 students from over 150 campuses that were taken in 2020, 65% of students reported not purchasing textbooks because of their high costs ([VCU Libraries Research Guides](#)). Our project BookCycle is a web application intended for college students at the George Washington University who spend hundreds of dollars each semester on textbooks, only to never use them again in a few months once they are enrolled in new classes that require different textbooks. The truth is that if college students cannot gain access to the necessary and required textbooks, their academic success will take a hit, affecting their future. It is no secret that textbooks are expensive, but buying new or renting can have a financial strain on college students who have many other costs to account for as well. As demand increases for cost-effective solutions, students are faced with the difficult task of finding textbooks for cheaper or buying new textbooks, putting them under further financial strain as college students. Buying textbooks also harms the environment as they are only used for about 4 months a year and are often forgotten relics that students use to prop up their belongings in their college dorms.

BookCycle is unique in its design for many reasons. Users are only allowed to sign up to use the web application if they have a George Washington email account, which then has to be verified to establish a secure and trustworthy network of users only within our community. Once a student signs up with a verified email that is authenticated, they will then be asked what textbooks they are in need of and if they have any that they would be willing to enter into the system. Our matching algorithm will provide them with a feed of textbooks that matches their needs. Lastly, once they put in a request for the textbook, the book lister will fill out a form about the buildings and times they are available to make the exchange with the user. The user, after selecting and agreeing to a building and time slot, will then meet at the agreed time and place to receive the textbook. With this, all exchanges are done within our campus giving an added layer of safety, as well as avoiding the additional cost of shipping. Their textbook feed will be specifically curated to their needs based on what classes they are taking, but also pushing the

idea to re-enter the textbooks they got from another user back into the system when the semester is over for someone else to be able to use them the same way they did.

Community-based textbook sharing is what makes us different from our competitors. Websites such as Facebook Marketplace, CraigsList, and BookMooch, all charge their users to buy or rent their textbooks, in addition to paying a shipping fee. BookCycle has no shipping, no warehouse, no middleman, and no cost. All book swapping is done within a trusted zone on campus where both parties agree to meet at the same building and time on the George Washington campus to exchange the textbook. People value a product that comes from a member of their community and within a safe zone that they are already familiar with. The key differentiating factor of our product is that all of the users are students who do not wish that a lack of access to textbooks will negatively affect their education and academic success, and have the ability to operate in a safe area with one another while positively affecting their education, financial spending, and environmental impacts. Our system allows users to make these exchanges with other students who wish to pay it forward by putting up books they no longer have any use for into the system to make an impact and difference in our community.

The beauty of BookCycle is that it is designed to foster the growth of an ecosystem. Users can only “check out” 5 textbooks per semester which will encourage them to either donate more books they no longer have use for or return the ones they have used during the semester back cycling into the system. We aim to establish a network where the users will feel inspired to “pay it forward” to other students to further their academic success without inflicting further financial strain on top of already costly tuition. We want to encourage our users to recycle these textbooks back into the system and create a cycle that will close off that academic performance gap being negatively impacted by students who cannot afford to rent or buy textbooks each semester to use this platform to make a difference in our community.

Technical Summary

The technologies used for BookCycle are React for the User Interface, node.js for building the application, Express in node.js to build the web application framework for the backend applications, Material UI for the front-end framework for React, PostgreSQL for the database, TypeORM library to link the database, and AWS for hosting and authentication. What is novel about this project is that it operates on a unique trusted zone within the GW campus and community, which we will be implementing through our web application. It is technically novel that the application and exchange itself will all be based on the Gale-Shapely algorithm, whereas existing platforms will have users manually search for their textbooks.

Our key objectives to be accomplished during this project is to have the user successfully sign up with a GW email account, have the user be able to successfully add a book in the database that they are willing to donate, have the user be matched with a textbook using the algorithm and get a list of textbooks, have the user be able to request an exchange, have the user fill out a form detailing the time and location of the exchange, have the users agree on an arranged meeting with the lister and exchange the textbook, and finally having the user fill out a short survey after the transaction to rate the exchange. The questions we answer include “How will a student be matched with a textbook?” “How will a student's financial need be determined?” and “How will we communicate with other GW systems?”

Other than the existing tools and technologies mentioned earlier, we used APIs to build our project, such as for building the textbook postings, seeing what the available textbooks, the feed page, and signing up for your account with a verified GW login are. We are confident that all of this is doable because we used Trello, Slack, and a Gantt chart, as well as relied on our communication and team working skills for this project. We have discussed that if one team member feels stuck in the technical component they are responsible for, to reach out to the other team members to work together to solve the problem.

The development cost in terms of hardware was free. All team members have a laptop that they will use to develop the project. In terms of software development, all of the software is free and open-source except for AWS, which we have requested for. Pre-design we estimated that this will be about 1000 lines of code for a simple design that primarily focuses on functionality and being user-friendly, then as time permits make it more aesthetically pleasing for the beta release with the addition of more lines of code. The project milestones are to have

functioning buttons, a search textbook function, a scale for textbook quality, a post-exchange survey with a user rating, input controls, product/website navigation, containers, informational components about the project and our goal such as a “My Profile” and “About Us/Technology”, and clear icons to be user-friendly and communicate efficiently. The timeline for achieving these milestones is to work to build all of the basic functioning pages by January, then use the remaining months leading up to the beta release to enhance the UI.

Project Description

Use Cases

- User sign-in with verified GW email
- User donates a textbook
- User receives a list of matched textbooks upon request
- User fills out a form detailing time/place of exchange
- User successfully picks up a textbook

User Stories

As a student, I would like to sign up to use this platform so that I can receive a textbook from another student on campus.

As a student, I would like to enter my courses, so that I can be matched with textbooks based on their ISBN.

As a student, I would like to donate my old textbooks on this platform, so that I can help another student in need.

As a student, I would like to request a book after being matched, so that I am able to exchange them.

As a student, I would like to set the dates, times, and places I am available to give my textbook to another student.

As a student, I would like to agree on what dates, times, and locations I can pick up the textbooks so that I am able to use them.

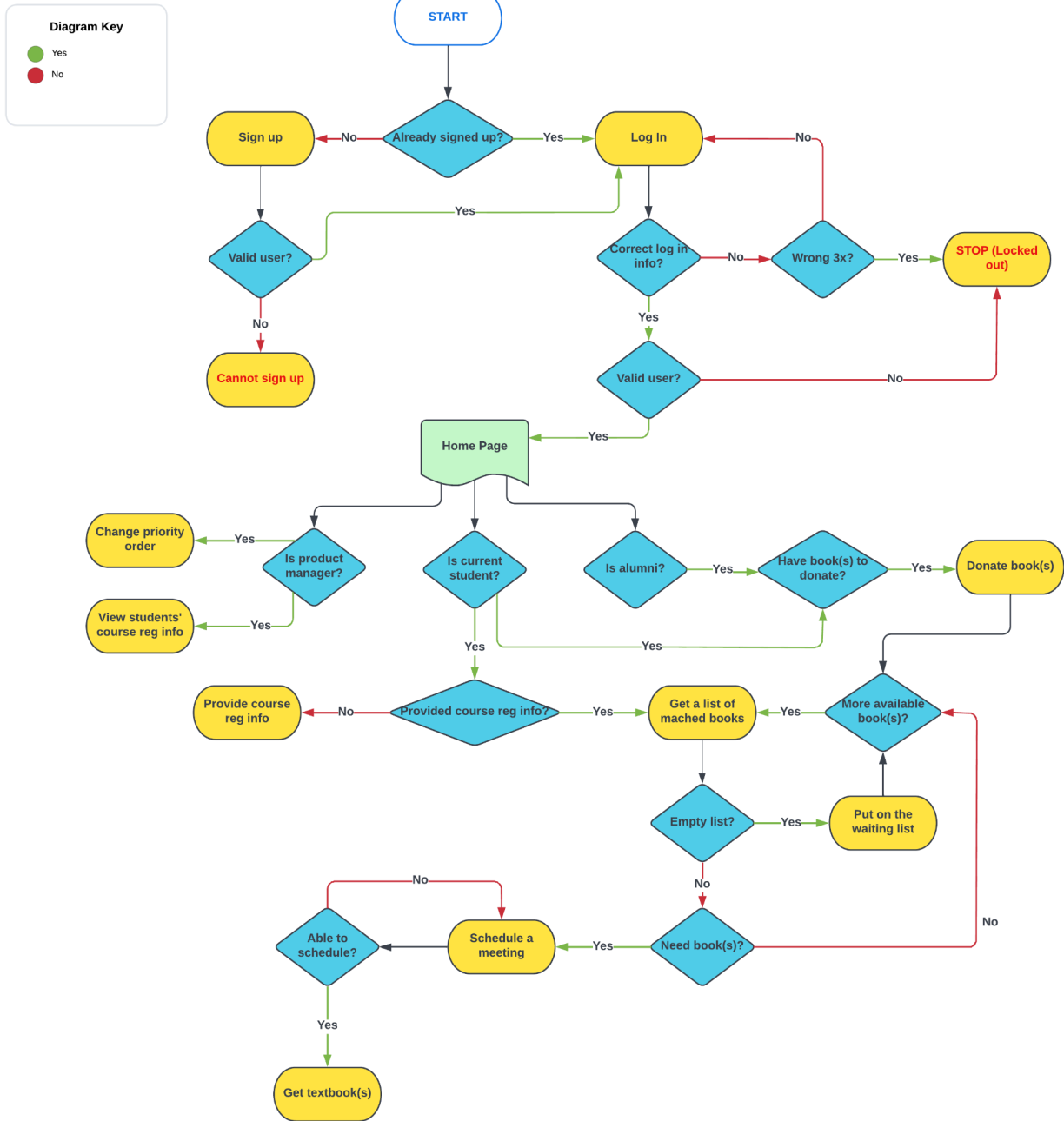
As a student, I would like to meet with another student after agreeing on a time, location, and date, so that I can use my textbook.

As a product manager, I would like to see a student's course registration information, so that the algorithm can match them with textbooks.

As a product manager, I would like to change the priority order so that I can adjust the system policy.

As a product manager, I would like to update the course information (e.g., books linked to a course) so that the right books will be matched to students.

Flow Diagram



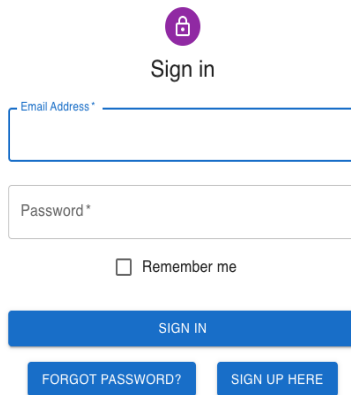
Mockups/Wireframes:

- API authentication
 - Server side sessions
 - Store session data in memory on the server in a session table

- Session ID -> session data

- Store session IDs in cookies on the client's browser
- Make sure the cookies are HTTP only and Same Site cookies to protect against CSRF and possible XSS attacks
- /api/register -> set cookie and implement session
- /api/login -> check credentials and implement session

- Login/Signup

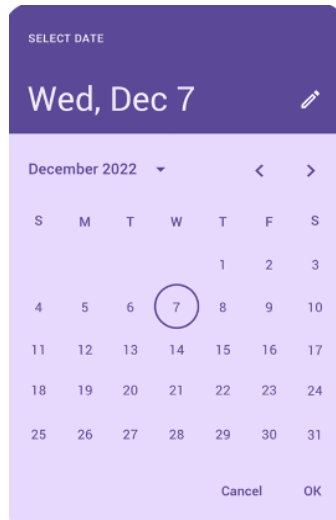


A sign-in form with a purple lock icon at the top. Below the icon is the text "Sign in". There are two input fields: "Email Address*" and "Password*". Below the password field is a checkbox labeled "Remember me". At the bottom are three buttons: "SIGN IN", "FORGOT PASSWORD?", and "SIGN UP HERE".

Copyright © CSJ 2022.

○

- Select meetup date



A date picker interface with a purple header "SELECT DATE" and "Wed, Dec 7". Below the header is a calendar for "December 2022". The days of the week are labeled S, M, T, W, T, F, S. The date "7" is circled. At the bottom are "Cancel" and "OK" buttons.

○

- Textbook Icon



○

- Create post icon



○

- Example textbook feed

Im a feed item

Lots of fun here

○

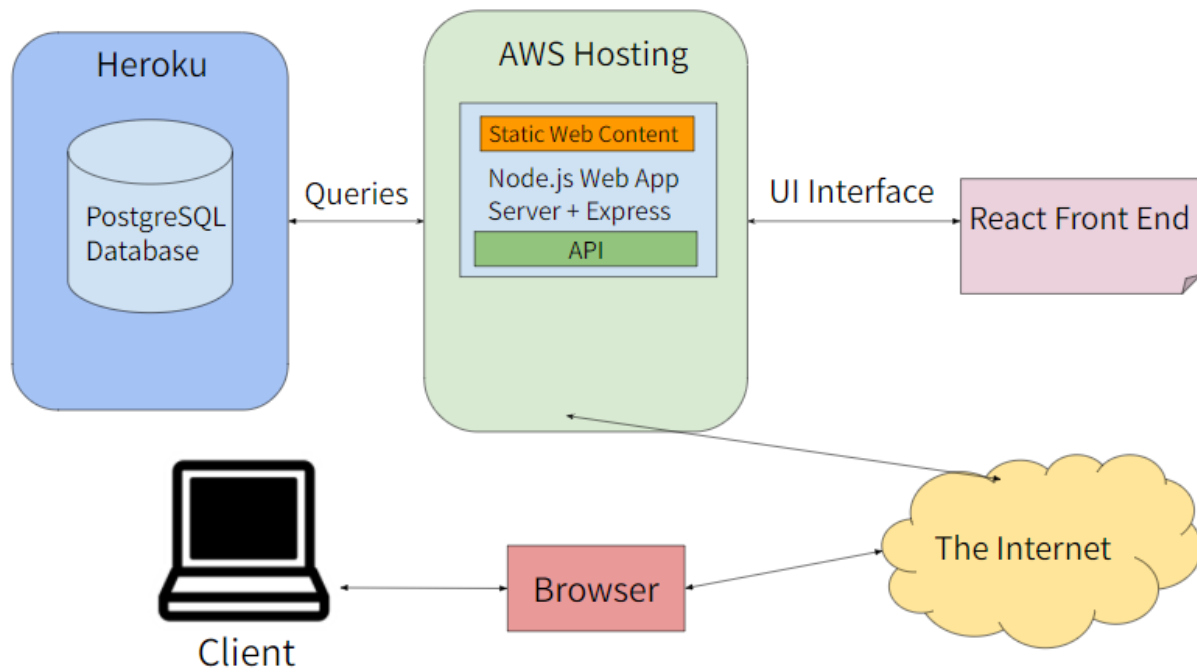
PRIMARY

SECONDARY BUTTON

Technical specifications:

Architecture/System Diagrams:

System Architecture



External APIs and Frameworks: This should be a list of all of the external APIs and frameworks that you call on or use to build your project. Each item should have a detailed description of why and how it is used in your project.

Name: React

Goal: Streamline the front-end development process to create a better UX

Description: React is a popular front-end framework that helps front-end developers create interactive UI interfaces.

Name: Material UI

Goal: Utilize pre-built components to create a good-looking user interface.

Description: This framework/library will be used to establish frontend components such as a navigation bar, containers, drop-down menus, selection components, and more.

Name: Heroku

Goal: Host our backend API and PostgreSQL database

Description: Deploy our backend server to Heroku so our frontend application can communicate with it efficiently. Additionally, Heroku can host our database so our API can retrieve data

Name: Express

Goal: Serve our API endpoints.

Description: Develop routes for each API endpoint that serves data from our database. Additionally, develop an authentication system using server-side sessions where session data is stored in memory on the server and the session id is stored in an HTTP-only cookie on the client's browser

Name: Axios

Goal: Send requests to our API.

Description: Exchange data from the client to the server using HTTP requests.

Name: PostgreSQL

Goal: Store data in a relational database

Description: Store important user information and textbook data that can be ultimately used in the application.

Name: TypeORM

Goal: Safely retrieve and store data in our PostgreSQL database from our backend server

Description: Create and alter models from our Node.js backend to store and retrieve data for users.

Algorithms

Matching algorithm

Goal: The goal of this algorithm is to match students with textbooks based on their financial needs given their course registration information.

- When demand is high but supply is low, users who are in financial need will be prioritized to get books with higher costs from a list of books that they need—this is the default setting.
- The system manager can adjust the priority order if the current system policy needs to be modified.
- The success of running this algorithm means that a student in financial need will be matched with a textbook based on their course registration information and mock financial need status.

Description:

- Using mock data with the GW financial aid office, students will be able to enter their GW id which then links with their financial aid to display them a priority score from 1-5 based on their financial need.
- The system will create 5 ranges of financial need, with 5 being the most financial need, so the algorithm will know who to prioritize.
- Given a list of factors to consider, which include prices of books, students' financial need status, number of books matched, number of books exchanged, and number of books donated, the system manager can update the priority order. The default setting will prioritize students who are in financial need the most.
- The system will create preference lists: a preference list of books for each student and a preference list of students for each course subject (book). Both preference lists will be sorted according to the current priority order.
 - For each student, the system will create an array object which stores the list of books that a student needs. Using the default setting, an array will be ordered according to the prices of books.
 - The system will create a 2D-array object that stores a list of students who will be taking a course for each course subject. The system will use appropriate queries to get a list of students, in which students are sorted into multiple levels based on priority order. Using the default setting, the system will use the following priority order: students' financial need status, the number of books exchanged, the number of books matched, and the number of books donated.

- Once the system has preference lists, it will apply the Gale-Shapley Algorithm, an algorithm for the stable marriage problem, as this algorithm is used to find a stable match between two sets given certain preferences. Note that the time complexity of the algorithm is $O(N^2)$.
 - The system will iterate through all the course subjects. Each course will iterate through a list of sorted students who have not been matched with associated textbooks and match students with available books. If some pair exists, it will also check whether this textbook is preferred over the previously matched textbook or not. If so, the student will be rematched. The system will repeat this process to find stable pairs for students and available textbooks.
- By entering their Gwid, students will be automatically matched with suitable textbooks given their course information.