Team CSJ - Writing 3

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.

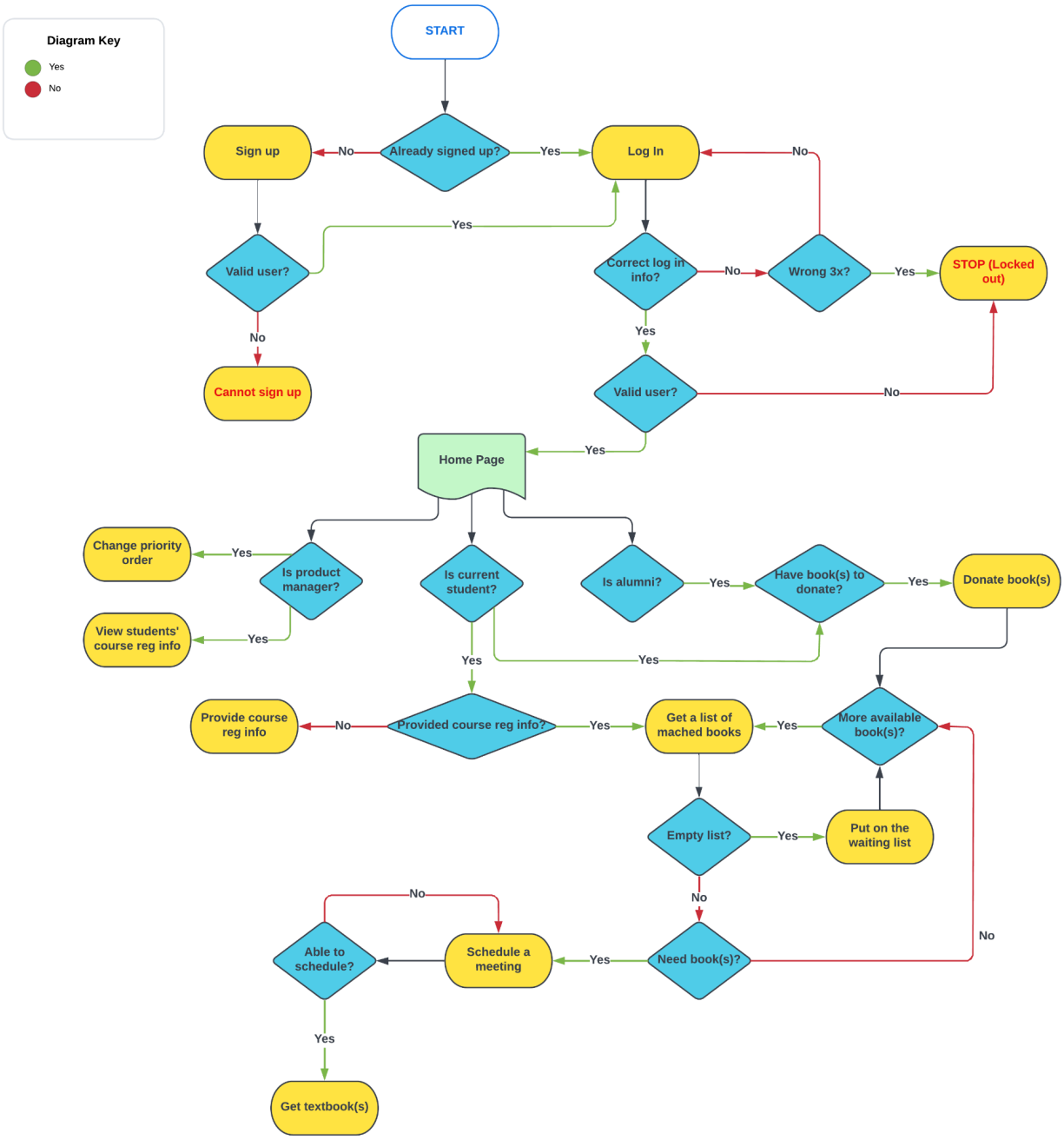
As a student, I would like to meet with another student after agreeing on a time, location, and date, so that I can use my textbook.

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.

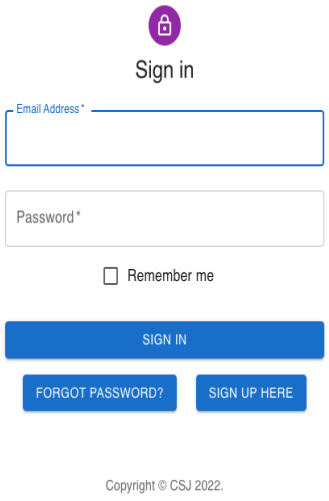
Flow Diagram



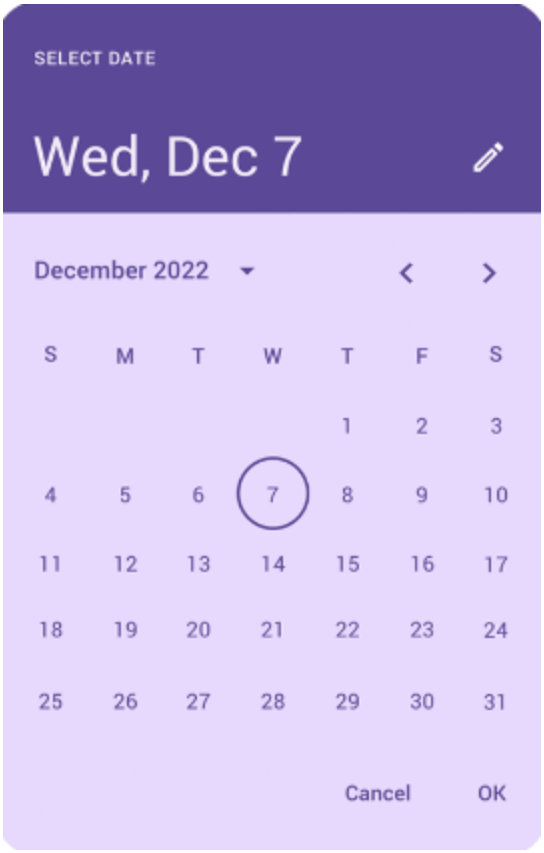
Mockups/Wireframes: What are the visual artifacts of what the end user will see when using your product/system? If you have a UI component that drives the user experience, this section should be images of those UI screens. If you have an API component that drives the user experience, this section should document the API UX: authentication setup (if applicable), list of all the API endpoints, code examples, and details on how to use each endpoint.

- 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

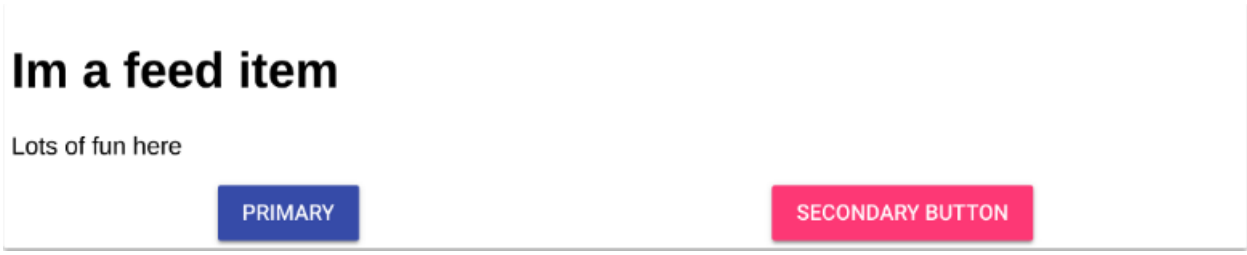


  -
- Select meetup date



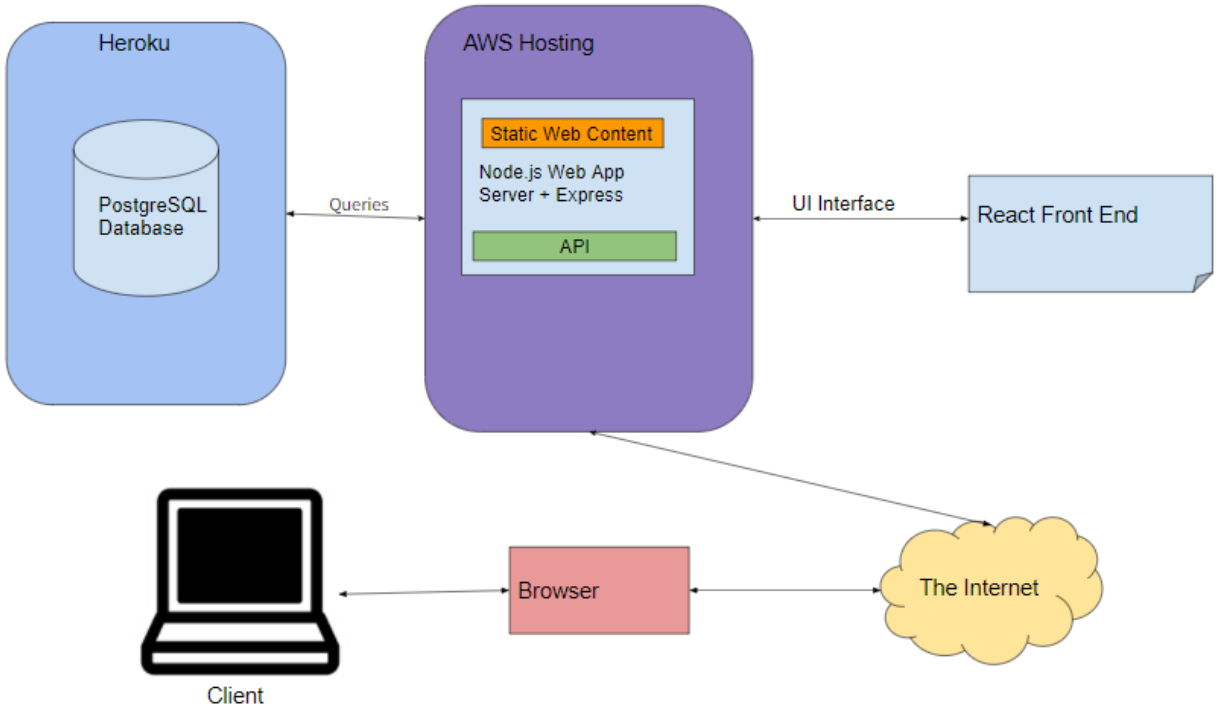
- Textbook Icon

- Create post icon

- Example textbook feed
  - 

Technical specifications:

Architecture/System Diagrams: How do all of the individual components of the project interact with each other? Now that you have each described your respective components of the project, this section should be the highest level view of how all of those components interact with each other.



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 frontend development process to create a better UX

Description: React is a popular frontend framework that helps frontend 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 they 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 need 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 donated, 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 prices of books.
    - The system will create a hashmap which stores a list of students who will be taking a course for each course subject. The system will use appropriate query to get a list of students, in which students are sorted in multiple levels based on the 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 have 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. For each course, it 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 more 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.
- Through entering their GWid, students will be automatically matched with suitable textbooks given their course information.