The Hof Book Nook

Architecture Design

Version: 2.0

Date: 11/30/22

Team:

Brian Baptista, Christopher Rios, Michael Salomone, Jasmin Varela

Table of Contents

I. Introduction

The purpose of this document is to provide details on how the previous development plans and requirements will be delivered by the end of the project. This is done by providing a closer look at The Hof Book Nook from an architectural standpoint.

II. Implementation & Configuration Management

The Hof Book Nook is being developed in Flutter. There are currently two versions of the Hof Book Nook app. One version of the app used a Bloc design pattern. The other version of the code did not use the Bloc design pattern. Due to delays in development, the project will continue to be developed without the design pattern.

In order to develop it we are using VS Studio Code and Android Studio Code. To test our functions, the program is run using emulators through Android Studio Code. This includes running it on web browsers such as Chrome and Edge for quick tests, for example checking UI formats, and on Android phone emulators for greater functions such as signing into the account.

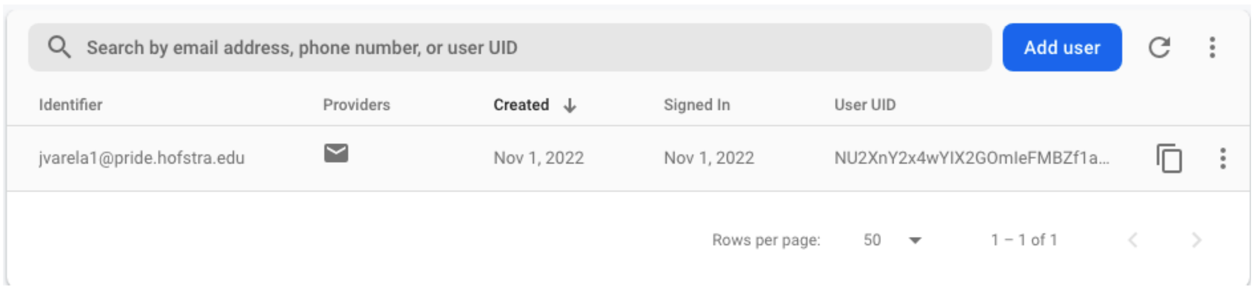
For our database we are using Firebase to store login information, user information, and books for sale. We will also be using Firebase to connect to our API (application programming interface). The API we are using is Google Books API. The API will provide information such as the textbooks' title, author, description and cover page. In addition to the Google Books API, the emailjs API is used for the email function of the app.

For our configuration management, we are using GitHub. This allows us to all be able to update and merge functions to test their functionality. Since we changed the way we are designing the project, the new link for the repository will be https://github.com/JVarela02/the_hof_book_nook.

III. Database Information

As stated previously, Firebase will be used as our database storage. Within Firebase, we will be using Firebase Authentication and Firestore.

Firebase Authentication will be used to store the user's login information (email and password). Firebase Authentication automatically hashes the passwords, which eliminates an extra task for us while at the same time guaranteeing the safety of the user's information. Additionally, Firebase Authentication provides us with beneficial built-in functions such as logging in with email and password and a reset password function. From the console point of view, we can see the following …

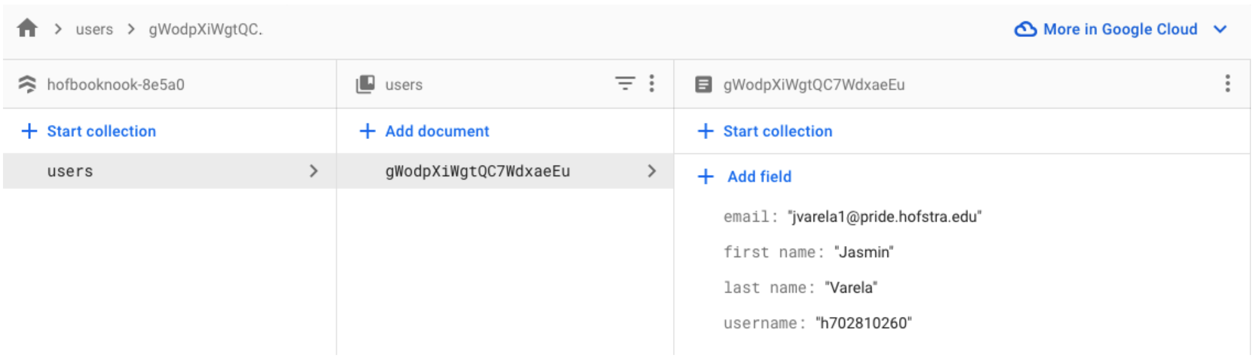


Identifier	Providers	Created ↓	Signed In	User UID
jvarela1@pride.hofstra.edu	✉	Nov 1, 2022	Nov 1, 2022	NU2XnY2x4wYIX2GOmleFMBZf1a...

From here, we can eliminate a user account or disable it.

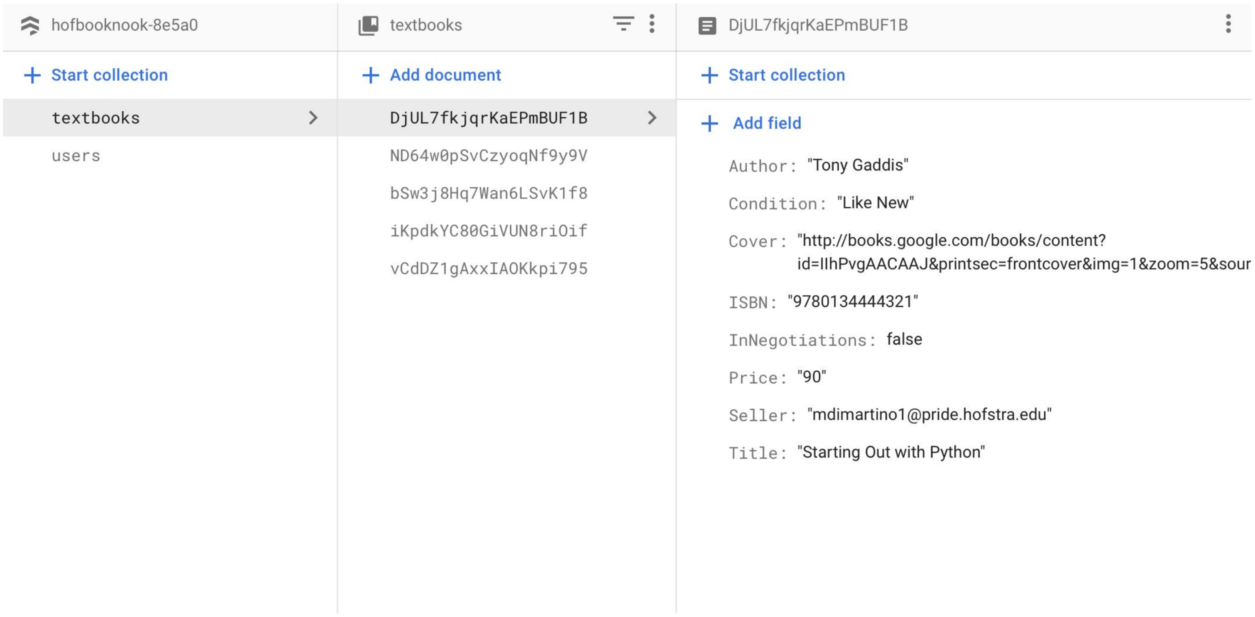
Firestore will be used to store the user data and the textbooks that are listed for display. In Firestore, we will use collections to keep the user information separate from the textbook information. In the "users" collection, we will store the user's first name, last name, h700#, and email address. This data will be used to log in, find textbooks for sale related to that user for my listing, delete textbook, and add textbook functions.

The "textbooks" collection stores the ISBN number, condition, and price of the textbook inputted by the user. Before sending the data to Firestore, the system connects to the API router and obtains the textbook's author, description, and link for the cover page. The user and API information is collected and stored in one single document.



users > gWodpXiWgtQC.

hofbooknook-8e5a0	users	gWodpXiWgtQC7WdxaeEu
+ Start collection	+ Add document	+ Start collection
users	gWodpXiWgtQC7WdxaeEu	+ Add field
		email: "jvarela1@pride.hofstra.edu"
		first name: "Jasmin"
		last name: "Varela"
		username: "h702810260"



hofbooknook-8e5a0	textbooks	DjUL7fkjqrKaEPmBUF1B
+ Start collection	+ Add document	+ Start collection
textbooks	DjUL7fkjqrKaEPmBUF1B	+ Add field
users	ND64w0pSvCzyoqNf9y9V	Author: "Tony Gaddis"
	bSw3j8Hq7Wan6LSvK1f8	Condition: "Like New"
	iKpdkYC80GiVUN8riOif	Cover: "http://books.google.com/books/content?id=IIhPvgAACAAJ&printsec=frontcover&img=1&zoom=5&sour
	vCdDZ1gAxxIAOKkpi795	ISBN: "9780134444321"
		InNegotiations: false
		Price: "90"
		Seller: "mdimartino1@pride.hofstra.edu"
		Title: "Starting Out with Python"

IV. Architecture Overview

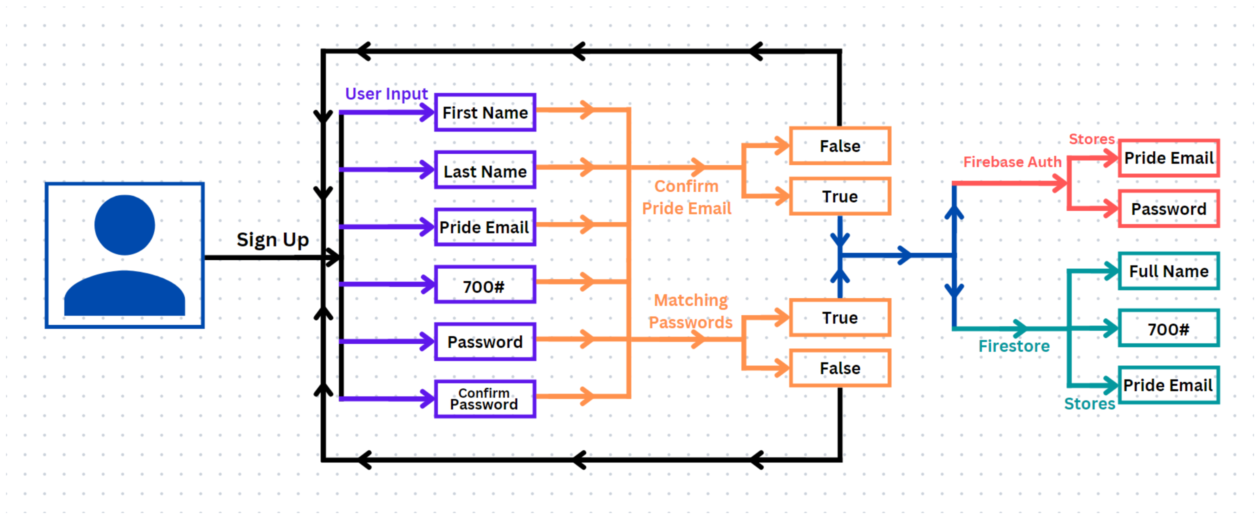
The project will follow a repository architectural design pattern. Everything displayed will be grabbed from the Firebase database to display book listings on the main page and keep track of authentication.

Using a repository design schema works well for our project as it allows us to easily manage all listings made on the servers as well as update listings, whether they are in negotiations or they have been sold, through a simple function call to the database. Using this design pattern also allows us to easily retrieve information from the Firebase Database without reducing performance issues by sifting through a list of all current listings, trying to grab all the appropriate information that needs to be displayed. The database will store information about users, such as their username, first name, last name, and email address. While the other part of the database, which controls textbooks, will keep track of the condition, if it is in negotiations, the seller's email address, the ISBN, and the book's price currently listed.

Authentication will also be run through the database to ensure that users have an account or have added an account with the correct credentials. The server will constantly make calls to the database to grab information to display, and the database will be able to grab information from the Google Books API to get book information that is not filled out.

V. Function Architecture

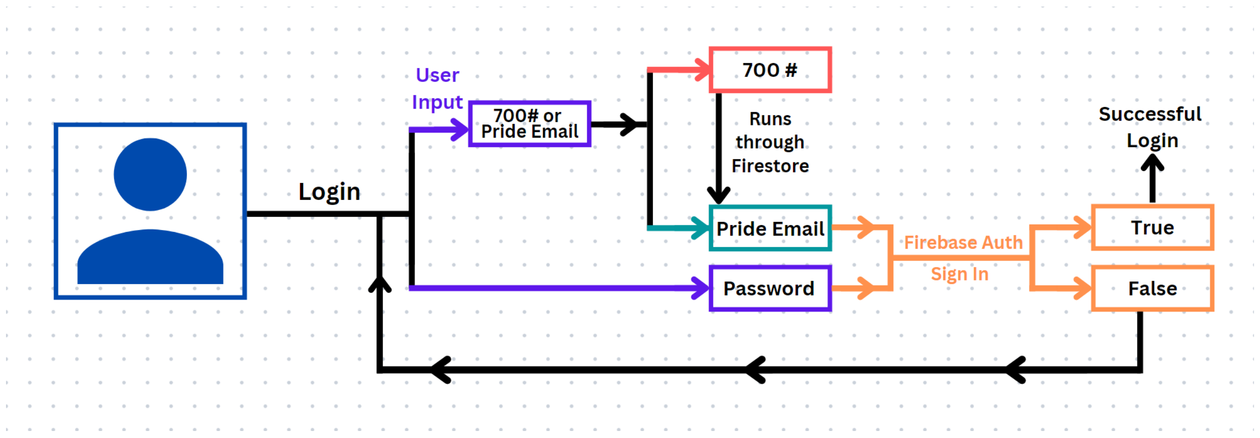
a. Sign-Up Function



When a new user wants to use the Hof Book Nook, they will have to create an account. On the sign-up page, they will be asked to submit their first and last name, pride email, 700#, and password twice. This information will be collected as text editing controllers. When the user attempts to create an account, the system will check for two things. The first thing will be that the password and confirmed password inputs match. If the passwords do not match, the user will be presented with an alert dialogue that will inform the user that the passwords do not match. The second thing the system will check is that the email the user is trying to submit is a Pride email address. This will be accomplished by checking if the email string includes "@pride.hofstra.edu." Like the password confirmation, if the email is not a Pride email, the user will receive an alert dialogue that informs them to change the email address.

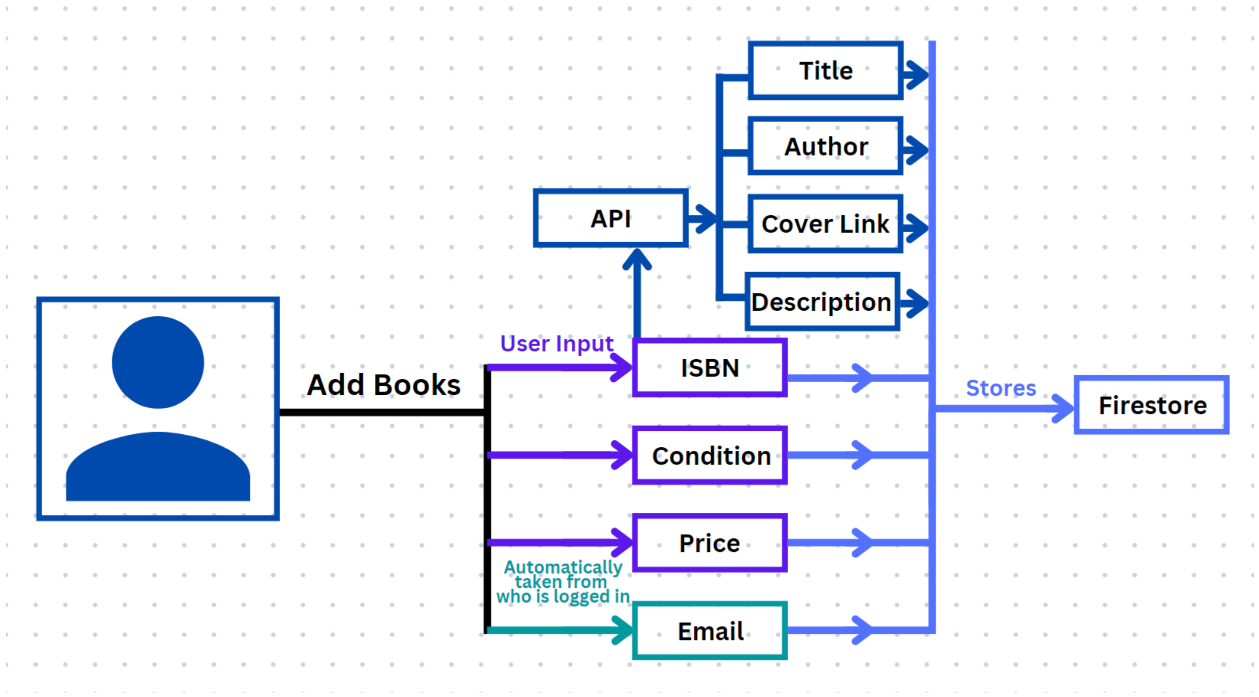
Once both tests are confirmed, the data is split into two storages. The Firebase Authorization will take the email and password. Firebase Auth will be in charge of allowing users to login in the future and run the "forgot password" if implemented.

b. Login Function

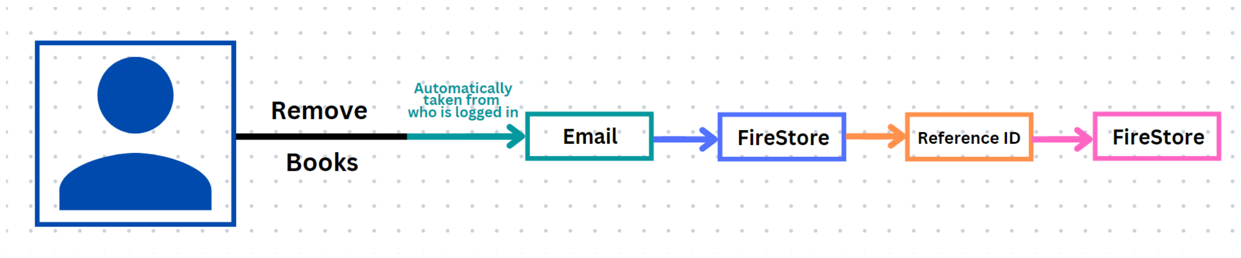


The login function works in a similar function to the sign-up function. The first thing it will do is check if the input is a 700# or pride email address in our database. If the 700# number matches one within our database, it will run through Firestore to the matching email address. The second thing the system will check is if the password matches the password in the database associated with that email address. If the password does not match, the user will be presented with an alert dialog informing them the password is incorrect.

c. Add and Remove Books

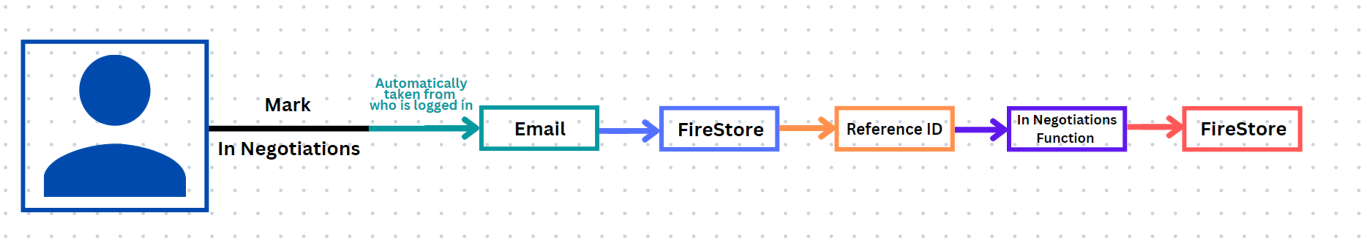


“Add books” works similarly to the “sign-up” function. It takes note of the email the user is logged in with and stores the email address with the information the user inputted about the book. This way, even if there are multiple of the same book, it will still have a personal identifier for the user selling it. Additionally, even if a user is selling multiple of the same book, each book for sale will have its own reference ID. In combination with this, the system also takes the imputed ISBN number and runs it through the API to get the title of the textbook being sold, its description, author, and cover link.



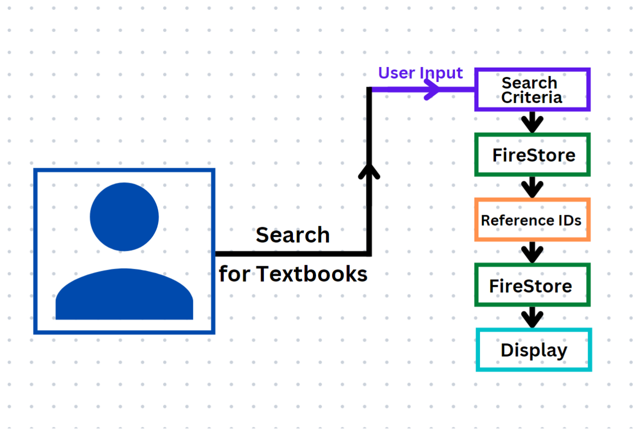
“Remove Books” works similarly to the “search for textbook” function. It uses the logged-in user’s email to find the reference ID from Firestore of books that have a seller with that email. It’ll collect the reference IDs of all matching books. When a user selects a textbook from the list, it’ll take the reference ID and call for Firestore to delete the book.

d. Mark in Negotiations



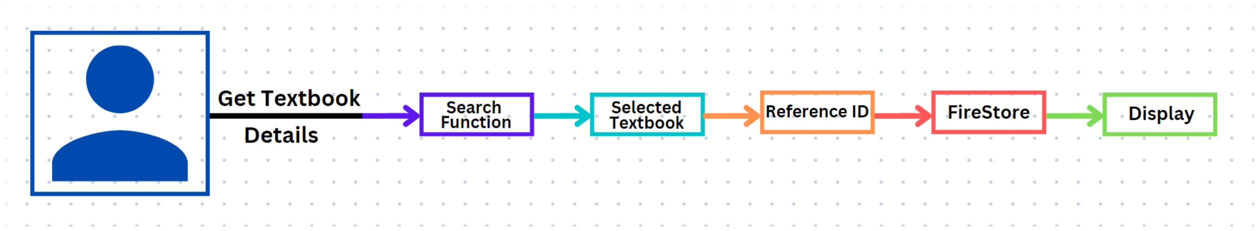
This function works similarly to “delete textbook.” It will take the email for the user logged in to search for matching sellers in Firestore. If a user selects a book, it’ll take the reference ID of the selected book and search for it in Firestore again. Once found, it will change the boolean field of “in negotiation” to true if false and false if true in Firestore.

e. Search for Book



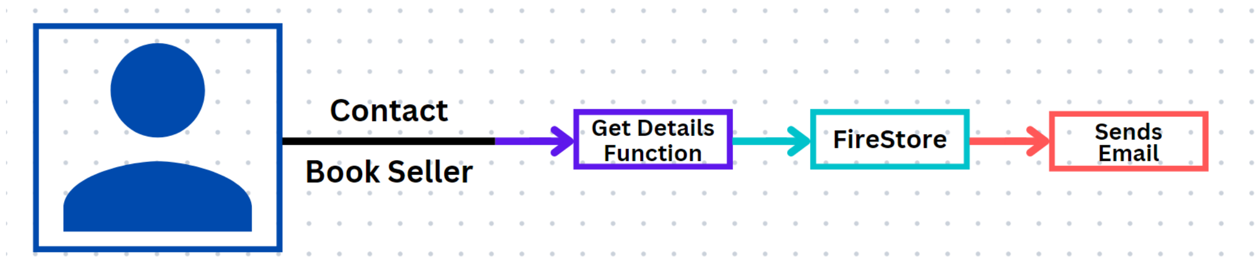
The system takes the user input and searches through the system. The search uses a dropdown button on the homepage to differentiate what is being searched, either an author, ISBN, or book title. To avoid parsing through the database multiple times, the user must also select the type of search that they want to be performed. This way, when the system searches for the ISBN, title, or author, it'll only have to search through that type. Once the reference IDs of matching textbooks are found, it'll once again pass through the database to collect the rest of the information of the textbook and display the results.

f. Get more information on the Textbook



This function is one of the simpler functions in terms of data flow. After the search function, a user will be able to select a textbook and will be given additional information about the textbook. From here, the system will use the reference ID of the selected textbook to search through the database for additional information. This information includes the title, author, and description of the textbook. Once the information is collected, it will be displayed back to the user.

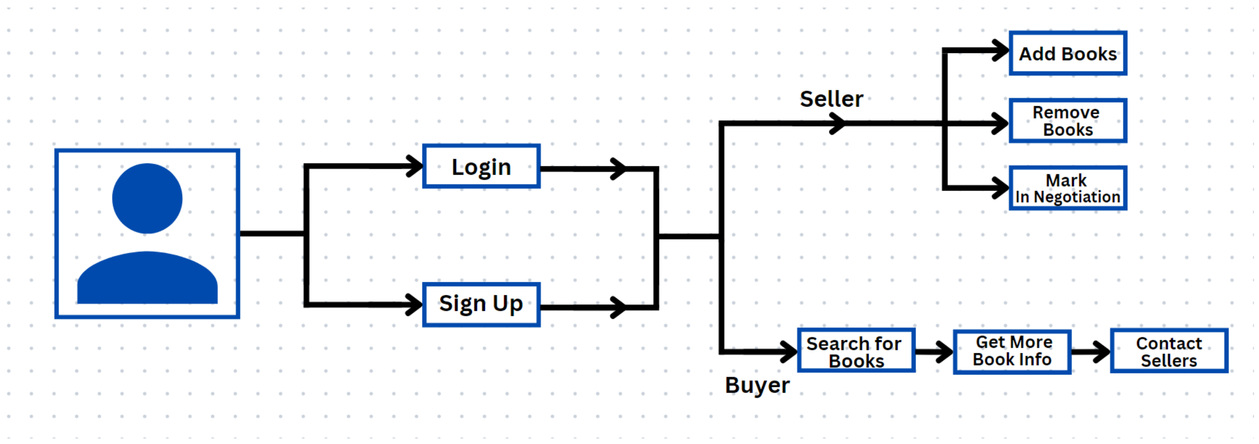
g. Contact Seller



The contact seller function builds on top of the "Get more information" function. When the function collects the additional information of the textbook it is also collecting the information of the seller of the textbook. This is then used in the email function which is built using emailjs API. The other information that is being collected from the database is the name and email address of the interested buyer, the user signed in, so the email sent to the buyer is also included so they can continue communicating with each other.

VI. User Access Flow

Below is a depiction of how users will be able to interact with the application and how this meets the requirements that were stated in the Requirements Specification Document.



When a user opens an application, they will be taken to either a login or signup page if they have not signed in before or if they've logged out. If a user is given a login page, but they don't have an account, they will be able to toggle to the signup page and vice-versa. From here, there will be three routes that their interaction with the app can take.

If a user wants to use the seller functions, they will be able to add books for sale, remove them from the database when transactions are complete, or choose to no longer sell, and to be able to mark a book as "in negotiations."

From a buyer's standpoint, the main function that they will be able to access is to search for books. From there, they will be able to access the requirements of getting more information on the textbook, and what we access from the API, and they will be able to contact the seller to purchase a book.

VII. Progress

Our original plan of attack for this project was as follows (the dates being our expected completion dates for each task).

1. Create & Connect Database - October 20, 2022
2. Create GUI - October 20, 2022
3. Create Functions for Sign-Up and Log-in - October 27, 2022
4. Create Functions for Add Books - October 27, 2022
5. Create Function to Remove Book - October 27, 2022
6. Create My Listings Display Function - November 5, 2022
7. Create Function to Mark "In Negotiations" - November 5, 2022
8. Create Search Function - November 14, 2022
9. Create Textbook Display Function - November 21, 2022
10. Create Notify Seller / Help Function - November 21, 2022

Due to delays, we've had to reschedule our project plan in order to have the functions implemented without impacting our testing & quality time. The way we've come to the following plan is by reallocating who's working on what task, temporarily putting task 7 on the back burner, and rescheduling our intended milestone dates. Our new schedule, not including the task we already completed, is as follows …

1. Create Functions for Add Books
2. Create Function to Remove Book
3. Create My Listings Display Function
4. Create Search Function
5. Create Get More Info. on Textbook Display Function
6. Create Notify Seller / Help Function

Tasks 1 and 2, purple text, must be completed by November 8, 2022

Tasks 3 and 4, blue text, must be completed by November 16, 2022

Task 5 and 6, green text, must be completed by November 24, 2022

For each of these milestones we will perform implementation and testing on the dates listed above. This will still allot us one week of testing before the final presentation date.