



# **SPU BOOK FINDER**

Prepared for: Sayra Flores

Proposed by Team Jakku  
Celena Flores, Monica Guevara, Tianxing Wang,  
and Cesar Labastida



## Executive Summary

---

Our client, Sayra Flores, on behalf of students at Seattle Pacific University has contacted Team Jakku to plan, implement, and maintain a new system that will be beneficial to all students at Seattle Pacific University (SPU). The proposed system, SPU Book Finder, will be a web-based application to assist SPU students in the process of purchasing and selling used textbooks.

After careful consideration of the System Proposal Request, we have decided the proposed system is feasible and low risk given the team members' experience and knowledge in data structures, application programming, net-centric programming, and database management. The team consists of four members: Celena Flores (team leader), Monica Guevara (recorder), Tianxing Wang, and Cesar Labastida.

This document highlights the unified vision of Team Jakku and Sayra Flores to build a web application that is more organized and efficient for students buying and selling used textbooks. The following document will consist of the team information, project description, system services description, nonfunctional requirements, development and operating environments description, data structural design, behavioral design, user interface design, and the development plan.



## Team Information

---

**Team Name:** Jakku

**Team Members and Contact Information:**

1. Celena Flores
  - a. floresc1@spu.edu
  - b. Team Leader
2. Monica Guevera
  - a. guevaram@spu.edu
  - b. Team Recorder
3. Tianxing Wang
  - a. wangt1@spu.edu
4. Cesar Labastida
  - a. labastidac@spu.edu

# Table of Contents

<b>1.0 PROJECT DESCRIPTION</b>	<b>1</b>
1.1 SYSTEM DESCRIPTION AND RATIONALE	1
1.2 SYSTEM SCOPE	1
1.3 SYSTEM DEVELOPMENT CONSTRAINTS	2
<b>2.0 SYSTEM SERVICES</b>	<b>3</b>
2.1 FUNCTIONAL REQUIREMENTS	3
2.3 USE-CASE DIAGRAM	5
2.3 USE-CASE DESCRIPTIONS	7
<b>3.0 NONFUNCTIONAL REQUIREMENTS</b>	<b>14</b>
3.1 OPERATIONAL REQUIREMENTS	14
3.2 PERFORMANCE REQUIREMENTS	14
3.3 SECURITY REQUIREMENTS	14
3.4 CULTURAL AND POLITICAL REQUIREMENTS	14
3.5 PROCESS REQUIREMENTS/ PROJECT CONSTRAINTS	14
<b>4.0 DEVELOPMENT AND OPERATING ENVIRONMENTS</b>	<b>15</b>
4.1 SYSTEM ARCHITECTURE	15
4.2 HARDWARE ENVIRONMENT	15
4.3 SOFTWARE ENVIRONMENT	15
4.4 SYSTEM SECURITY	16
4.5 DEVELOPMENT ENVIRONMENT	16
<b>5.0 DATA(STRUCTURAL) DESIGN</b>	<b>17</b>
5.1 CLASS DIAGRAM	17
5.2 DATA DICTIONARY (METADATA)	18
<b>6.0 BEHAVIORAL DESIGN</b>	<b>31</b>
6.1 SYSTEM SERVICES IMPLEMENTATION MODEL	31
6.2 STATE CHART DIAGRAM(S)	34
<b>7.0 USER INTERFACE DESIGN</b>	<b>35</b>
7.1 USER INTERFACE REQUIREMENTS AND CONSTRAINTS	35
7.2 WINDOW NAVIGATION DIAGRAM	36
7.3 SCREEN INTERFACE DESIGN	37
<b>8.0 DEVELOPMENT PLAN</b>	<b>44</b>
8.1 DEVELOPMENT SCHEDULE	44

# 1.0 Project Description

---

## 1.1 System Description and Rationale

---

The system, SPU Book Finder, will be a web-based application that will facilitate the process of purchasing and selling textbooks. The resources students are currently using include a Facebook page, the SPU Bookstore, and Amazon.com. The Facebook page is an invite only group with about two-thousand members that allows users to post a short description and/or pictures of the textbooks. This style of organization is difficult to navigate and control because it is hard to search for a specific book and know the accurate availability of the book. The SPU Bookstore allows users to purchase or rent used textbooks, however, if there are none available students will inevitably have to purchase or rent the textbook at retail value. The SPU Bookstore, Amazon.com, and other websites are well organized, but lack in saving students time and money. Thus, SPU Book Finder will provide students with an organized platform where students can purchase and sell their textbooks at lower, comparable prices.

## 1.2 System Scope

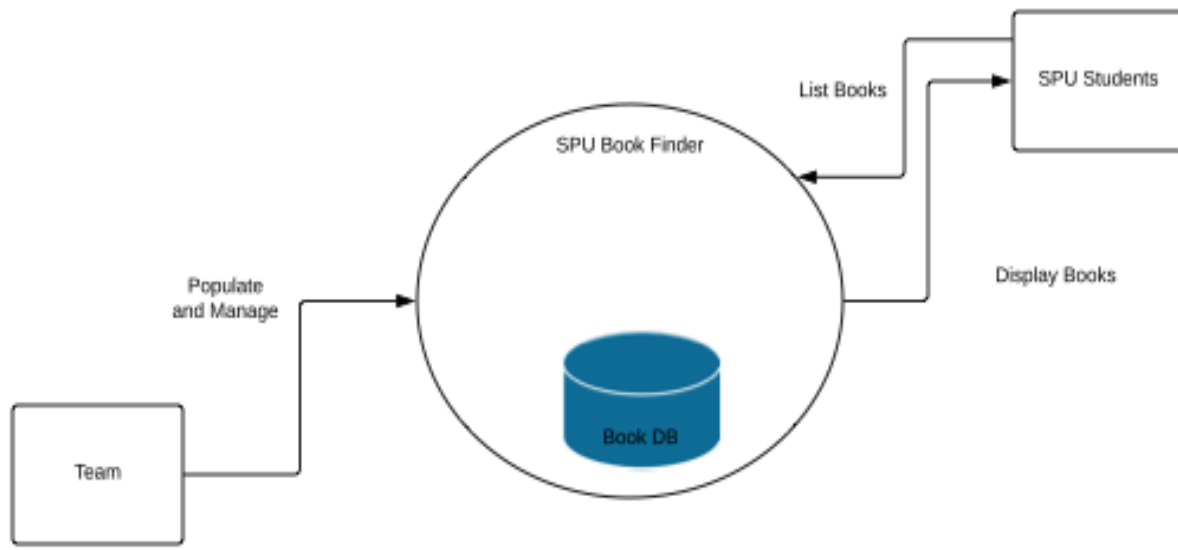
---

The following list includes the major requirements of SPU Book Finder:

- The system must provide users with a simple form to list their textbook.
- The system must allow users to search for a textbook based on ISBN, title, or class information (course number and department).
- The system must display a list of available textbooks to users.
- The system must provide a safe method for a buyer to contact a vendor by providing a security measure before receiving contact information.
- Initially the system will be populated with textbooks from the CS department and be readily adaptable to textbooks of other departments at Seattle Pacific University.
- The system will not perform any payment transactions on the web application.

### Context Diagram:

Below there is a context diagram for SPU Book Finder. A context (free) diagram illustrates the entities the system will come in contact with and the basic relationships between them and the system. The diagram below shows two entities and the interaction they have with SPU Book Finder. The first entity, is our team, which will populate and manage the system. The second entity, are any outside users that interact with our system. In our case, it will only be authorized students at SPU. Students at SPU will be able to list books into the system and retrieve information out of the system. The information they will get back is any available used textbook relevant to the search.



### 1.3 System Development Constraints

---

The following list includes potential constraints and solutions that may arise in the SPU Book Finder's operation, development and usage:

- SPU Book Finder needs access to Seattle Pacific University's class and book database.
  - Our team will mitigate this by obtaining the information off the Seattle Pacific University's course catalog and populating the information in our own database.
- SPU Book Finder needs users to list textbooks to be successful.
  - Our team will advertise our system throughout the school by using flyers and posting on various social media platforms.
- SPU Book Finder will need resources, such as a server, to support all web application functionalities.
  - Our team will mitigate this by using free available online resources provided to students like Amazon Web Services, Digital Ocean, or Heroku.
- SPU Book Finder will need to prevent fraudulent and pirated selling of textbooks.
  - Our team will mitigate this by providing a system that does not allow textbooks to be listed unless the ISBN of textbooks is in our database. Additionally, vendors will have to agree not to post fraudulent and pirated textbooks prior to listing. Lastly, we will make sure that users that sign up for our service have an email with SPU to keep outside fraudulent users outside the system.

## 2.0 System Services

---

The section will describe the following for SPU Book Finder: its functional requirements, a use-case diagram, and their corresponding use-case descriptions. Functional requirements state or describe a system process or service. Functional requirements are related and can be traced to a specific use-case. Use-cases are the common scenarios and user interactions with a system. They will be displayed together in a use-case diagram using the Unified Modeling Language (UML). UML is a modeling language in software engineering with the purpose of providing a standardized way to visually represent the design of the system. Lastly, each use-case will have a complimentary use-case description. Use-case descriptions provide the detailed steps, relationships, and outcomes of each individual use-case.

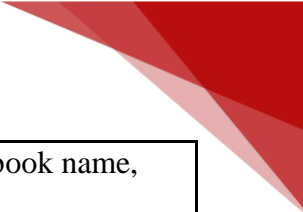
### 2.1 Functional Requirements

---

<b>Create Account</b>	The system must allow SPU students to create an account
Use-Case(s):	1
Initial Acceptance Tests:	<ul style="list-style-type: none"><li>• The user must use their SPU email to create an account</li><li>• Once an account is created the user must receive an email in which he/she confirms their account</li><li>• Once the email is confirmed the user can access the SPU Book Finder system</li></ul>

<b>Posting New Book</b>	The system must allow a user to post their book for sale and edit or delete the posting
Use-Case(s):	2, 3, 4
Initial Acceptance Tests:	<ul style="list-style-type: none"><li>• Attempt to post with missing information Name, ISBN, or SPU class number missing Book condition and price missing Seller contact information missing Book picture is missing</li><li>• Attempt to post a book that is not part of the SPU curriculum or database</li><li>• Edit a posting that has been deleted</li></ul>



<b>Textbook Search</b>	The system must allow users to search for textbooks by typing the book name, ISBN number or SPU class number in the search bar.
Use-Case(s):	5
Initial Acceptance Tests:	<ul style="list-style-type: none"> <li>• Name of the book entered does not return a match</li> <li>• ISBN entered does not return a match</li> <li>• SPU class number does not return a match</li> <li>• Name of the book entered returns more than one match</li> <li>• ISBN entered returns more than one match</li> <li>• SPU class number returns more than one match</li> <li>• If all three categories are entered: <ul style="list-style-type: none"> <li>But the name of the name of the book is incorrect</li> <li>The ISBN number is incorrect</li> <li>Or the SPU class number is incorrect</li> </ul> </li> <li>• User searches for a book posting that has been sold</li> </ul>

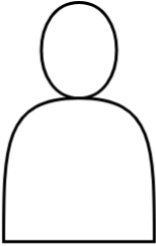
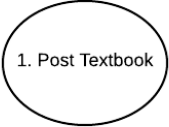



<b>Sorting Search Result</b>	The system must allow a user to sort their search result based on price, book condition or relevance.
Use-Case(s):	6
Initial Acceptance Tests:	<ul style="list-style-type: none"> <li>• Sort results by price <ul style="list-style-type: none"> <li>Lowest to highest</li> <li>Highest to lowest</li> </ul> </li> <li>• Sort results by book condition <ul style="list-style-type: none"> <li>New</li> <li>Used-Like New</li> <li>Used-Good</li> <li>Used-Acceptable</li> </ul> </li> <li>• Sort results by relevance <ul style="list-style-type: none"> <li>New listing</li> </ul> </li> <li>• Sort results when a combination of conditions are selected (Price: lowest to highest and Condition: Used-Good)</li> <li>• Sort search results when only a single book is found</li> <li>• Sort search results when no books were found</li> </ul>

<b>Communication</b>	The system must allow buyers to communicate with sellers through their SPU email (Since communication is going to be performed through SPU email a disclaimer needs to be provided)
Use-Case(s):	7
Initial Acceptance Tests:	<ul style="list-style-type: none"> <li>• The email used by the seller or buyer, should be the confirmed email used when the account was created</li> </ul>

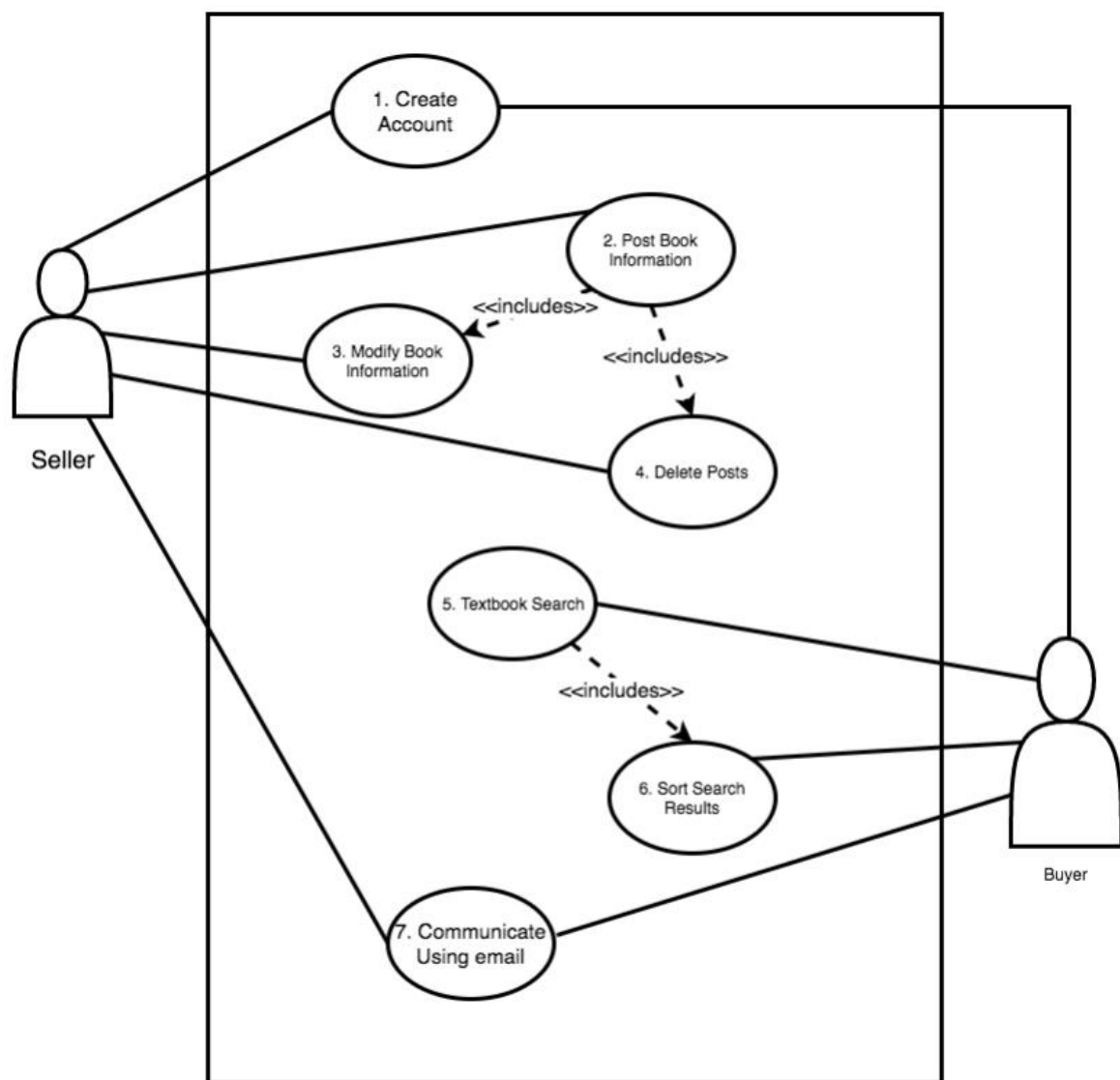


## 2.3 Use-Case Diagram

For reference, the table below will explain the symbols represented in the use-case diagram.

	An actor is a user that interacts with the system. They represent a role a user can have when interacting with the system.
	A use-case represents different scenarios of SPU Book Finder. A number and name identify each use-case.
	A line or association is a connection between use-cases and actors. They show the interaction between an actor and a use-case.
	An include relationship represents the inclusion of the functionality of one use-case within another. They are drawn from the base use-case to the used use-case.
	An extend relationship shows the extension of the use-case to include optional behavior. It is drawn from the extended use-case to the base use-case.

## SPU Book Finder Use-Case Diagram



## 2.3 Use-Case Descriptions

<b>Use-Case name:</b> Create Account	<b>ID:</b> 1	<b>Importance:</b> High
<b>Primary actor:</b> Buyer and Seller	<b>Use-Case type:</b> Detailed, Essential	
<b>Stakeholders and interests:</b> Buyer and Seller want a secure way to access the system		
<b>Brief description:</b> This use-case describes the process of creating an account. By creating an account the system will be able to verify that the users are SPU students		
<b>Trigger:</b> Buyer or seller select "Create Account"  <b>Type</b> (circle one):    External		
<b>Relationships:</b> <b>Association:</b> Buyer and Seller <b>Include:</b> - <b>Extend:</b> - <b>Generalization:</b> -		
<b>Normal flow of events:</b> 1. User selects "Create Account" 2. The user is prompted to enter their information such as SPU email and a password 3. The user will receive an email prompting him/her to confirm their email 4. Once the user's email is confirmed they can access the system		
<b>Subflows:</b> None		
<b>Alternate / exceptional flows:</b> 4a: If the user attempts to access the system before confirming their email, access to the system should be denied		

<b>Use-Case name:</b> Post Book Information	<b>ID:</b> 2	<b>Importance:</b> High
<b>Primary actor:</b> Seller	<b>Use-Case type:</b> Detailed, Essential	
<b>Stakeholders and interests:</b> Seller: wants to sell their books and Buyer: wants to buy used textbooks		
<b>Brief description:</b> This use case describes the process of posting a book to the SPU Book Finder system.		
<b>Trigger:</b> Seller selects "Post New Book"  <b>Type</b> (circle one):    External		
<b>Relationships:</b> <b>Association:</b> Seller <b>Include:</b> 3. Modify book information 4. Delete post <b>Extend:</b> - <b>Generalization:</b> -		
<b>Normal flow of events:</b> <ol style="list-style-type: none"> <li>1. Seller selects "Post New Book"</li> <li>2. Seller will be taken to a page asking them to fill out information about their book.</li> <li>3. They must fill out: Book Title, Author's Name, book condition, a short description, price they want to sell it for, and provide a picture of the book.</li> <li>4. They review all the information in the review page and confirm their post.</li> <li>5. The post will now become public to all buyers.</li> </ol>		
<b>Subflows:</b> S-3: Verify that the book information was entered correctly		
<b>Alternate / exceptional flows:</b> 5a: If seller wants to change something in their post, perform use case 3. "Modify book information" 5b: After a book is sold, perform use case 4. Delete Post		

<b>Use-Case name:</b> Modify Book Information	<b>ID:</b> 3	<b>Importance:</b> High
<b>Primary actor:</b> Seller	<b>Use-Case type:</b> Detailed, Essential	
<b>Stakeholders and interests:</b> Seller: wants to change the book information when needed, most likely the descriptions and price in their post. Buyer: wants to have updated information about the books they are looking at.		
<b>Brief description:</b> Sellers will have the right to change some of their post information in case they need to lower the price or modify the book description.		
<b>Trigger:</b> The user selects “Modify Book Information” <b>Type</b> (circle one):    External		
<b>Relationships:</b> <b>Association:</b> Seller <b>Include:</b> - <b>Extend:</b> - <b>Generalization:</b>		
<b>Normal flow of events:</b> 1. The user selects “Modify Book Information” 2. The user updates the book information and selects “update”		
<b>Subflows:</b> S-1: Modify Book 1. If the user has more than one book they chose which one they want to modify S-2: Verfiy Book Information 1. The book information is checked to make sure the modified information is correct		
<b>Alternate / exceptional flows:</b> None		

<b>Use-Case name:</b> Delete Post	<b>ID:</b> 4	<b>Importance:</b> High
<b>Primary actor:</b> Seller	<b>Use-Case type:</b> Detailed, Essential	
<b>Stakeholders and interests:</b> Seller: wants to delete their post when they cannot sell the book anymore or after the book is sold. Buyer: wants to view the updated book inventory and only choose books that are available.		
<b>Brief description:</b> This use-case describes the process of deleting a post from SPU Book Finder.		
<b>Trigger:</b> The user selects “Delete Post” <b>Type</b> (circle one):    External		
<b>Relationships:</b> <b>Association:</b> Seller <b>Include:-</b> <b>Extend:-</b> <b>Generalization:-</b>		
<b>Normal flow of events:</b> 1. The user selects “Delete Post” 2. The seller goes to the particular post and deletes the corresponding post 3. The user is then prompted why they would like to delete the posting 4. The book inventory is now updated.		
<b>Subflows:</b> S-3: Delete Post 1. The user has the option of selecting that the post is being deleted because they sold the book or just delete 2. If the book was sold the system stores the book sold under the user’s account so that they have a record of the books they have sold and the book post is removed 3. If the user just wants to delete the post the post is removed or deleted from SPU Book Finder		
<b>Alternate / exceptional flows:</b> None		

<b>Use-Case name:</b> Textbook Search	<b>ID:</b> 5	<b>Importance:</b> High
<b>Primary actor:</b> Buyer	<b>Use-Case type:</b> Detailed, Essential	
<b>Stakeholders and interests:</b> Buyer: wants an efficient way to search for available books Seller: wants to be noticed or found by the buyer		
<b>Brief description:</b> This use case describes the process of searching SPU Book Finder for available books		
<b>Trigger:</b> Buyer selects the search bar in the SPU Book Finder system  <b>Type</b> (circle one):    External		
<b>Relationships:</b> <b>Association:</b> Buyer <b>Include:</b> 6. Sort Search Results <b>Extend:</b> - <b>Generalization:</b> -		
<b>Normal flow of events:</b> 1. User enters the book information into the search bar or selects the book information from the drop-down menu 2. The system displays all available books that fall under the user's search criteria 3. The user can then select the book they are interested in to learn more about it		
<b>Subflows:</b> S-1: The user can enter either the book name, ISBN number, or SPU class number		
<b>Alternate / exceptional flows:</b> 2a: If the user would like to sort the search results then perform use case 6. Sort Search Results		

<b>Use-Case name:</b> Sort Search Results	<b>ID:</b> 6	<b>Importance:</b> High
<b>Primary actor:</b> Buyer	<b>Use-Case type:</b> Detailed, Essential	
<b>Stakeholders and interests:</b> Buyer: wants a way to sort the search results Seller: wants an efficient way to be found by buyers		
<b>Brief description:</b> Allowing buyers to sort search results will benefit both sellers and buyers. Buyers can find the books they want more easily which allows sellers to sell their books in a shorter time.		
<b>Trigger:</b> Buyer selects filter search results  <b>Type</b> (circle one):    External		
<b>Relationships:</b> <b>Association:</b> Buyers <b>Include:</b> - <b>Extend:</b> - <b>Generalization:</b> -		
<b>Normal flow of events:</b> 1. Buyer searches for a book 2. A list of results is shown to the Buyer 3. The Buyer can sort the search results		
<b>Subflows:</b> S-3: Sort Search Results 1. Results can be filtered by price, book condition, and relevance		
<b>Alternate / exceptional flows:</b> None		



<b>Use-Case name:</b> Communicate Using Email	<b>ID:</b> 7	<b>Importance:</b> High
<b>Primary actor:</b> Seller and Buyer	<b>Use-Case type:</b> Detailed, Essential	
<b>Stakeholders and interests:</b> Buyer: wants a way to communicate with the seller Seller: wants a secure way to be contacted by the buyer		
<b>Brief description:</b> This use-case describes the process of a buyer contacting a seller through email.		
<b>Trigger:</b> Buyer selects “Contact Seller”  <b>Type</b> (circle one):    External		
<b>Relationships:</b> <b>Association:</b> Buyer <b>Include:</b> - <b>Extend:</b> - <b>Generalization:</b> -		
<b>Normal flow of events:</b> 1. The user selects “Contact Seller” 2. The user is able to email the seller by contacting him/her through their email 3. The user can now contact the Seller through email		
<b>Subflows:</b> None		
<b>Alternate / exceptional flows:</b> None		

## 3.0 Nonfunctional Requirements

---

The following section describes the nonfunctional requirements of SPU Book Finder. Nonfunctional requirements are described as the characteristics and attributes of a system as well as its boundaries and limitations. The operational requirements of the system will give the operating environment in which the system must operate. Performance requirements tend to describe the system's reliability and response time. Security requirements describe the ability to protect the information system from intentional or unintentional data loss. Cultural and political requirements are requirements that are specific to the country in which the system functions. Process requirements or project constraints describes any constraints in the development process.

### 3.1 Operational Requirements

---

1. The system must be accessible using Windows, Mac, Android, or iPhone using either
  - a. Internet Explorer, Google Chrome, Safari, or Firefox
2. The system must be able to import pictures (These will be used when posting a book)

### 3.2 Performance Requirements

---

1. The web page must load in 10 seconds or less
2. The system must process book posts in under 10 seconds
3. The system must process textbook searches in less than 10 seconds
4. The available book database must be updated after a new book post has been added

### 3.3 Security Requirements

---

1. The system must only allow SPU students to create an account
2. The system must only allow users who have an account to post and find textbooks
3. A user's email and password must be encrypted for security
4. Only system administrators can have access to the book database

### 3.4 Cultural and Political Requirements

---

1. The system will only be available in English
2. After June 2018 the system should be crowdsourced (To ensure that the book database continues to be updated)
  - a. A minimum of three people will be required to validate any changes done to the database

### 3.5 Process Requirements/ Project Constraints

---

1. The system must provide minimal data entry
  - a. For example, when posting a new book the user only has to provide the book's title, ISBN, or SPU class number
2. The system must provide an organized user interface to ensure use of system
3. The system must provide messages
  - a. For example, acknowledgement messages when a book is successfully posted
4. The system must be consistent across different platforms
5. The system must have a connection to the internet

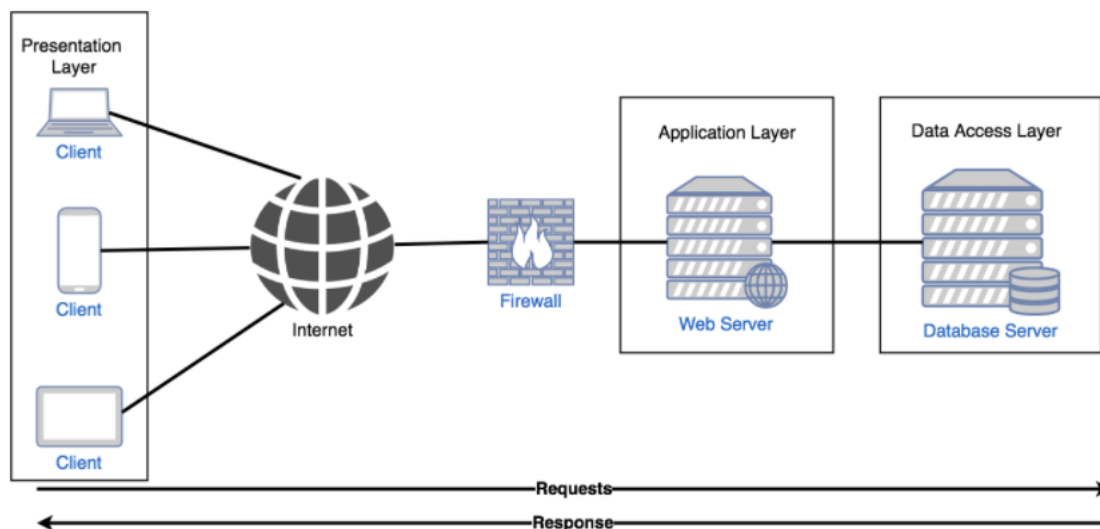
## 4.0 Development and Operating Environments

This Section outlines the hardware and software that are necessary for operating our system and gives an initial security plan. There will be an Architecture Overview Diagram showing what kind of hardware and software will be used in our system and how do they interact with each other.

SPU Book Finder will be a web-based interface, so the client device will be hosting the presentation logic of the web service and the web server will manage the data processing service.

### 4.1 System Architecture

SPU Book Finder will have a three-tier client server structure, in which the application server partitions tasks between the providers of the data service and service requesters. The students will use their device to request the server's data and get the results during their communication sessions with the server. The presentation tier translates the tasks and results to something the users can understand. The applications layer processes commands and make logical decisions. It also moves and process data between the two other layers. Whenever the user makes a request, the information in the data access layer will pass the requested information back to the application layer for processing and eventually back to the user.




### 4.2 Hardware Environment

Database, Web Application Servers must be purchased to store book inventory data and application logic. One main database server and one main web application server will include both database and application logic and should be present.

The requirements of Client device vary, but the recommended hardware is at least a 1.7 GHZ dual Core with 2 GB RAM.

### 4.3 Software Environment

The SPU Book Finder system requires an internet browser that supports websites coded using HTML5, CSS, JavaScript and PHP. Any modern operating system that supports one of the following internet browsers will enable a user to run the system. The supported browsers are



Safari 4 or newer, Firefox 3.5 or newer, Google Chrome, and Internet Explorer 9 or newer. These browsers are capable of supporting the system without additional cost.

#### 4.4 System Security

---

The Book Finder will only be accessible to authorized users, because the current version is open to the students at Seattle Pacific University only. Thus, authenticate user identity is crucial for the security of the system. Besides, to lower the risk of fraud and hacking, there will be no online payment option in the current version. Buyers and sellers will process payment outside of our system.

#### 4.5 Development Environment

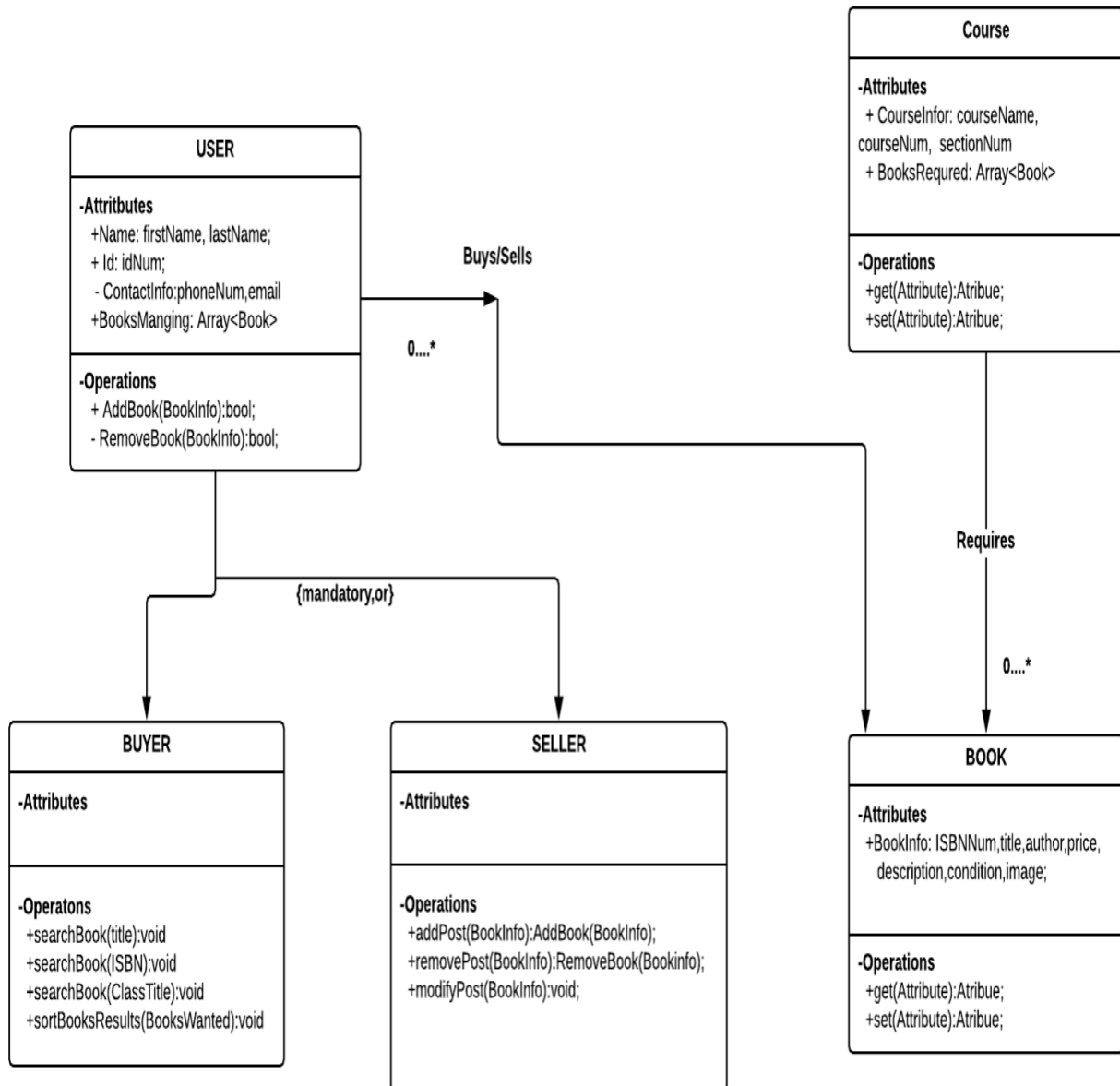
---

SPU Book Finder will be programmed in HTML5, CSS, JavaScript and PHP. All codes will be processed in sublime and Notepad++. Github Student Package will also be used during the development process.

## 5.0 Data(Structural) Design

A class diagram is shown to expose the details of SPU Book Finder system. A class diagram is useful to understand our systems functionality and data requirements. Also, this helps us better understand our system's inner dynamics. Additionally, we have included a Data Dictionary that will help understand each of the system's class and functions.

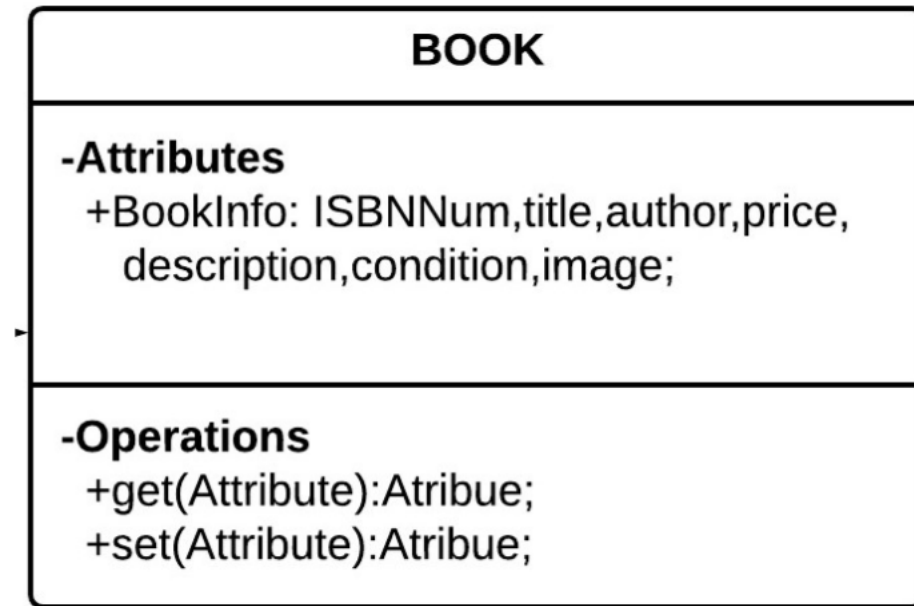
### 5.1 Class Diagram



## 5.2 Data Dictionary (Metadata)

---

### BOOK CLASS



**Name:** Book

**Visibility:** Public

**Documentation:** This class is in charge of representing each book in our system. This class will be used for querying and storing books in our database, and also for displaying book information in our User Interface thus it includes operations for both purposes.

**Attributes:** ISBNNum, title, author, price, description, condition, image.

ISBNNum -> ISBN Number of the book.

title -> Title of the book.

author - > Author or Authors of the book, this can be more than one.

price -> Price of the book.

description -> description of the textbook, here we can include the edition and any extra info of book.

condition -> condition of the book, new, used, etc.

image -> Image id to help us identify its picture.

#### **Operations:**

get(Attribute)-> get any attribute from instance book.

set(Attribute) -> set any attribute from instance book.

## ATTRIBUTES

Name	Type	Visibility	Multiplicity	Initial Value
ISBNNum	integer	Global	1	NULL
title	String	Global	1	NULL
author	String	Global	1...*	NULL
price	float	Global	1	0.0
description	String	Global	1	NULL
conditon	String	Global	1	NULL

## OPERATIONS

**getISBN();**

	<u>Name</u>	<u>Type</u>	<u>Visibility</u>	<u>isQuery</u>	<u>Default</u>
<u>Return</u>	ISBNNum	integer	global	true	0
<b>Parameters</b>	none	none	none	none	none

**setISBN(int ISBN);**

	<u>Name</u>	<u>Type</u>	<u>Visibility</u>	<u>isQuery</u>	<u>Default</u>
<u>Return</u>	none	none	none	none	none
<b>Parameters</b>	ISBNNum	integer	global	true	0

**getTitle();**

	<u>Name</u>	<u>Type</u>	<u>Visibility</u>	<u>isQuery</u>	<u>Default</u>
<u>Return</u>	bookTitle	String	global	<u>true</u>	Null
<b>Parameters</b>	none	none	none	none	none

setTitle(String bookTitle);

	<u>Name</u>	<u>Type</u>	<u>Visibility</u>	<u>isQuery</u>	<u>Default</u>
<u>Return</u>	none	none	none	none	none
Parameters	bookTitle	String	global	true	Null

getAuthorName();

	<u>Name</u>	<u>Type</u>	<u>Visibility</u>	<u>isQuery</u>	<u>Default</u>
<u>Return</u>	authorName	String	global	true	Null
Parameters	none	none	none	none	none

setAuthorName(String authorName);

	<u>Name</u>	<u>Type</u>	<u>Visibility</u>	<u>isQuery</u>	<u>Default</u>
<u>Return</u>	none	none	none	none	none
Parameters	authorName	String	global	true	Null

getPrice();

	<u>Name</u>	<u>Type</u>	<u>Visibility</u>	<u>isQuery</u>	<u>Default</u>
<u>Return</u>	price	float	global	true	0.0
Parameters	none	none	none	none	none

setPrice(float price);

	<u>Name</u>	<u>Type</u>	<u>Visibility</u>	<u>isQuery</u>	<u>Default</u>
<u>Return</u>	none	none	none	none	none
Parameters	price	float	global	true	0.0

getDescription();

	<u>Name</u>	<u>Type</u>	<u>Visibility</u>	<u>isQuery</u>	<u>Default</u>
<u>Return</u>	description	String	global	true	Null
Parameters	none	none	none	none	none





**setDescription(String description);**

	<u><b>Name</b></u>	<u><b>Type</b></u>	<u><b>Visibility</b></u>	<u><b>isQuery</b></u>	<u><b>Default</b></u>
<u><b>Return</b></u>	none	none	none	none	none
<b>Parameters</b>	decription	String	global	true	Null

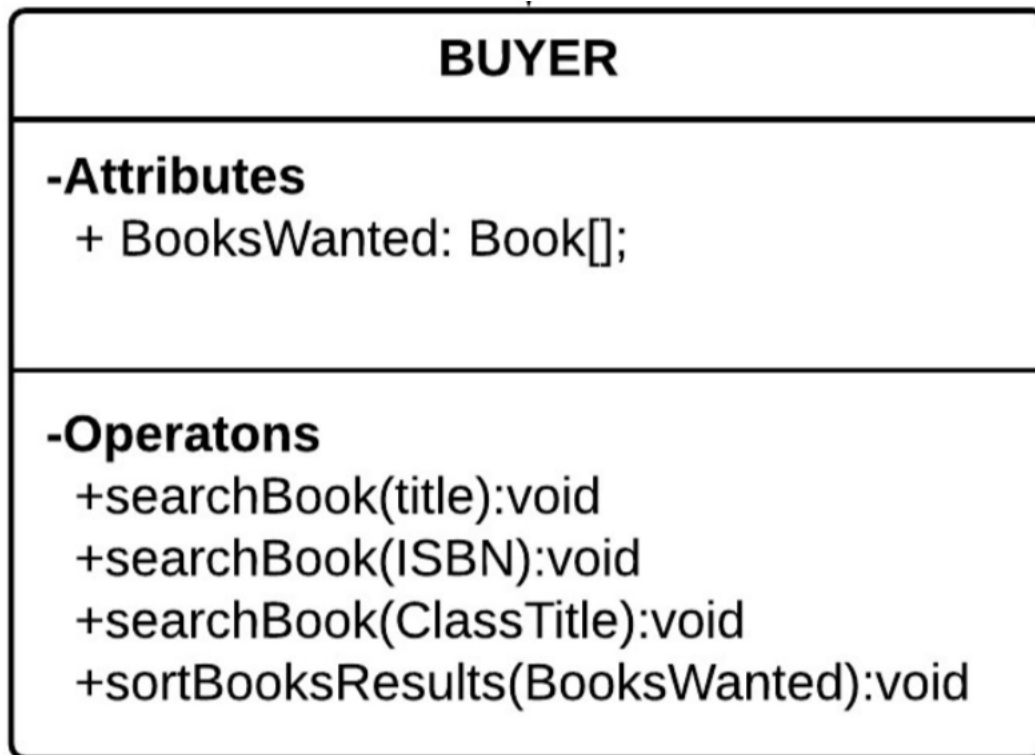
**getCondition();**

	<u><b>Name</b></u>	<u><b>Type</b></u>	<u><b>Visibility</b></u>	<u><b>isQuery</b></u>	<u><b>Default</b></u>
<u><b>Return</b></u>	condition	String	global	<u>true</u>	Null
<b>Parameters</b>	none	none	none	none	none

**setDescription(String description);**

	<u><b>Name</b></u>	<u><b>Type</b></u>	<u><b>Visibility</b></u>	<u><b>isQuery</b></u>	<u><b>Default</b></u>
<u><b>Return</b></u>	none	none	none	none	none
<b>Parameters</b>	decription	String	global	true	Null

## BUYER CLASS



**Name:** Buyer

**Visibility:** Public

**Documentation:** Buyer class is in charge of representing each buyer in our system. It is specifically design for user operation, thus we must implement operations. It will allow our buyers to store as many wanted books as they want, and also sort the results based on priority or price. Aside from inheriting all attributes and operations from USER class, BUYER class contains the following:

**Attributes:** BooksWanted.

BooksWanted -> an array of objects type book, this will allow us to store and sort any desire books from the user.

**Operations:**

searchBook(String title) -> perform a search using book title to determine if there are any sellers for this book.

searchBook(int ISBN) -> perform a search using ISBN number to determine if there are any sellers for this book.

searchBook(String ClassTitle) -> perform a search using Book title to determine if there are any sellers for this book.

sortBooks() -> sort the books wanted by user based on user preference or price.

## ATTRIBUTES

NAME	TYPE	Visibility	Multiplicity	Initial Value
booksWanted	Array of Books	Local	1	NULL

## OPERATIONS

### searchBook(String title);

	<u>Name</u>	<u>Type</u>	<u>Visibility</u>	<u>isQuery</u>	<u>Default</u>
<u>Return</u>	book	BOOK	global	<u>true</u>	Null
<u>Parameters</u>	title	String	global	true	Null

### searchBook(int ISBN);

	<u>Name</u>	<u>Type</u>	<u>Visibility</u>	<u>isQuery</u>	<u>Default</u>
<u>Return</u>	book	BOOK	global	<u>true</u>	Null
<u>Parameters</u>	ISBNNum	int	global	true	0

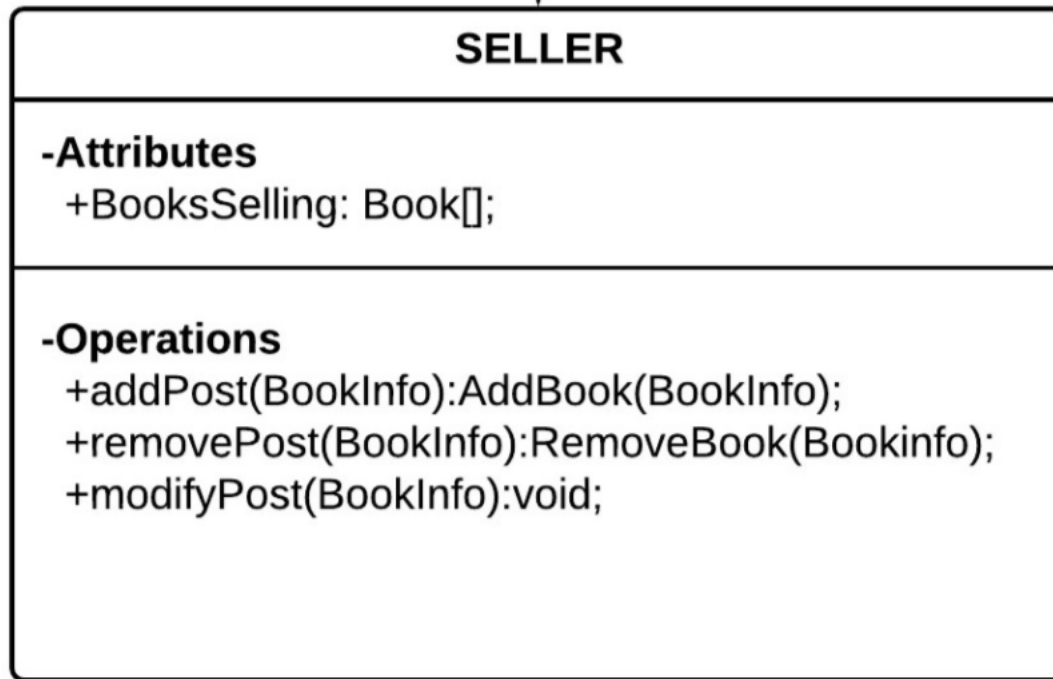
### searchBook(String courseTitle);

	<u>Name</u>	<u>Type</u>	<u>Visibility</u>	<u>isQuery</u>	<u>Default</u>
<u>Return</u>	book	BOOK	global	<u>true</u>	Null
<u>Parameters</u>	courseTitle	String	global	true	Null

### sortBooks()

	<u>Name</u>	<u>Type</u>	<u>Visibility</u>	<u>isQuery</u>	<u>Default</u>
<u>Return</u>	none	none	none	<u>none</u>	none
<u>Parameters</u>	none	none	none	none	none

## SELLER CLASS



**Name:** Seller

**Visibility:** Public

**Documentation:** This class is another type of implementation of user class. This class will be in charge of representing our sellers, the difference between BUYER and SELLER classes is that seller will be able to add posts or modify them. Aside from inheriting all attributes and operations from type USER class, SELLER will implement the following.

**Attributes:** BooksSelling

BooksSelling -> an array of objects type BOOK.

**Operations:**

addPost(BookInfo) -> adds a new post to the system, with a new book added to the database.

removePost(BookInfo) -> removes a post from the system, erases a book from the database.

modifyPost(BookInfo)-> uses getter and setters of BOOK class and updates the post.

## ATTRIBUTES

NAME	TYPE	Visibility	Multiplicity	Initial Value
booksSelling	Array of Books	Local	1	NULL

## OPERATIONS

**addPost(title,ISBNNum, author, price, decription, condition);**

	<u>Name</u>	<u>Type</u>	<u>Visibility</u>	<u>isQuery</u>	<u>Default</u>
<u>Return</u>	success	boolean	global	false	Null
Parameter1	title	String	global	true	Null
Parameter2	ISBNNum	int	global	true	0
Parameter3	author	String	global	true	Null
Parameter4	price	float	global	true	0.0
Parameter5	description	String	global	true	Null
Parameter6	condition	String	global	true	Null

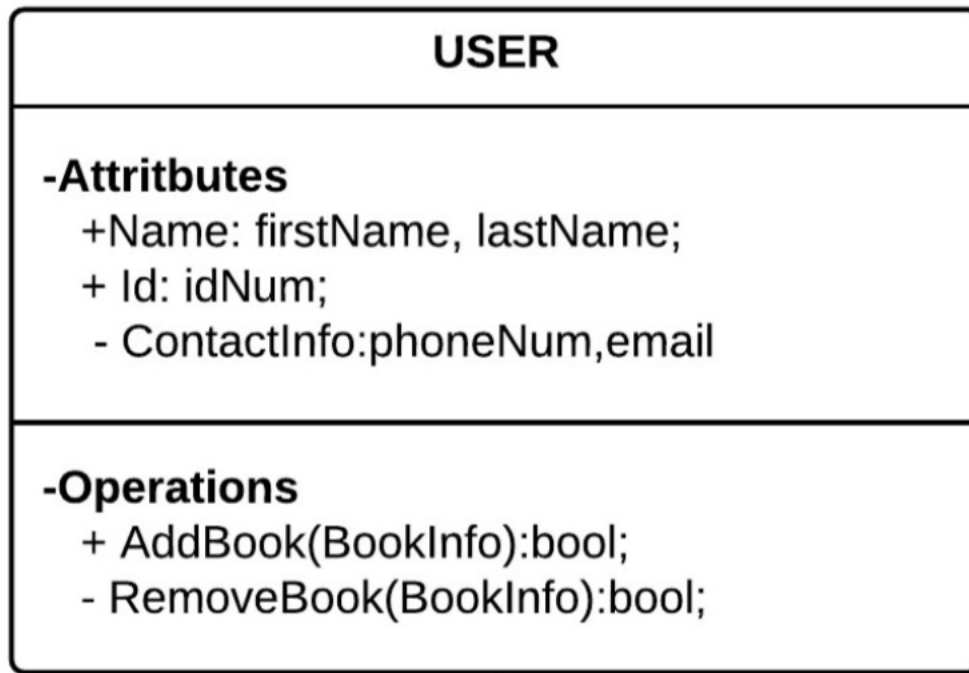
**removePost(BOOK selectedBook)**

	<u>Name</u>	<u>Type</u>	<u>Visibility</u>	<u>isQuery</u>	<u>Default</u>
<u>Return</u>	success	boolean	global	false	Null
Parameters	selectedBook	BOOK	global	true	Null

**modifyPost(BOOKATTRIBUTE atribue)**

	<u>Name</u>	<u>Type</u>	<u>Visibility</u>	<u>isQuery</u>	<u>Default</u>
<u>Return</u>	success	boolean	global	false	Null
Parameters	atribue	BOOK ATTRIBUTE	global	true	Null

## USER CLASS



**Name:** User

**Visibility:** Public

**Documentation:** USER class will be used as an abstract type class to implement any type of user in the system (SELLER or USER). Here we will store functions and attributes that USER And SELLER type users share in common.

**Attributes:** firstName, lastName

firstName -> First name of user.

lastName -> Last name of user.

idNum -> ID number to identify each user.

phoneNum -> phone number of user.

email -> email of user

**Operations:**

AddBook(BOOK book)-> adds book to the database.

RemoveBook(BOOK book)-> removes book from the database.

## ATTRIBUTES

NAME	TYPE	Visibility	Multiplicity	Initial Value
firstName	String	Local	1	NULL
lastName	String	Local	1	NULL
idNum	integer	Local	1	NULL
phoneNum	String	Local	1	NULL
email	String	Local	1	NULL

## OPERATIONS

### addBook(BOOK book)

	<u>Name</u>	<u>Type</u>	<u>Visibility</u>	<u>isQuery</u>	<u>Default</u>
<u>Return</u>	success	boolean	global	<u>false</u>	Null
<b>Parameters</b>	book	BOOK	global	true	Null

### removeBook(BOOK selectedBook)

	<u>Name</u>	<u>Type</u>	<u>Visibility</u>	<u>isQuery</u>	<u>Default</u>
<u>Return</u>	success	boolean	global	<u>false</u>	Null
<b>Parameters</b>	selectedBook	BOOK	global	true	Null

## COURSE CLASS

Course
<b>-Attributes</b> + CourseInfo: courseName, courseNum, sectionNum + BooksRequired: Array<Book>
<b>-Operations</b> +get(Attribute):Atribue; +set(Attribute):Atribue;

**Name:** Course

**Visibility:** Public

**Documentation:** Course class will allow us to store all the information for each class. This is so that we can display the books that are needed for a certain course. We also need the course name, course number and section number of the class so that we can help our users identify each class.

### Attributes:

courseName -> Course Name

courseNum -> Course Number

sectionNum -> Section Number

BooksRequired -> Array of books required

### Operations:

get(Attribute)-> get any attribute from course

set(Attribute) -> set any attribute from course



## ATTRIBUTES

NAME	TYPE	Visibility	Multiplicity	Initial Value
courseName	String	Local	1	NULL
courseNum	integer	Local	1	0
sectionNum	integer	Local	1	0
BooksRequired	Array<BOOK>	Local	1	NULL

## OPERATIONS

**getCourseName;**

	<u>Name</u>	<u>Type</u>	<u>Visibility</u>	<u>isQuery</u>	<u>Default</u>
<u>Return</u>	courseName	String	global	true	Null
Parameters	none	none	none	none	none

**setCourseName(String courseName);**

	<u>Name</u>	<u>Type</u>	<u>Visibility</u>	<u>isQuery</u>	<u>Default</u>
<u>Return</u>	none	none	none	none	none
Parameters	courseName	String	global	true	Null

**getCourseNumber();**

	<u>Name</u>	<u>Type</u>	<u>Visibility</u>	<u>isQuery</u>	<u>Default</u>
<u>Return</u>	courseNum	integer	global	true	0
Parameters	none	none	none	none	none



setCouseNumber(int coueseNum);

	<u>Name</u>	<u>Type</u>	<u>Visibility</u>	<u>isQuery</u>	<u>Default</u>
<u>Return</u>	none	none	none	none	none
<b>Parameters</b>	courseNum	integer	global	true	0

getCourseSection();

	<u>Name</u>	<u>Type</u>	<u>Visibility</u>	<u>isQuery</u>	<u>Default</u>
<u>Return</u>	courseSection	integer	global	true	0
<b>Parameters</b>	none	none	none	none	none

setCouseSectionr(int courseSection);

	<u>Name</u>	<u>Type</u>	<u>Visibility</u>	<u>isQuery</u>	<u>Default</u>
<u>Return</u>	none	none	none	none	none
<b>Parameters</b>	courseSection	integer	global	true	0

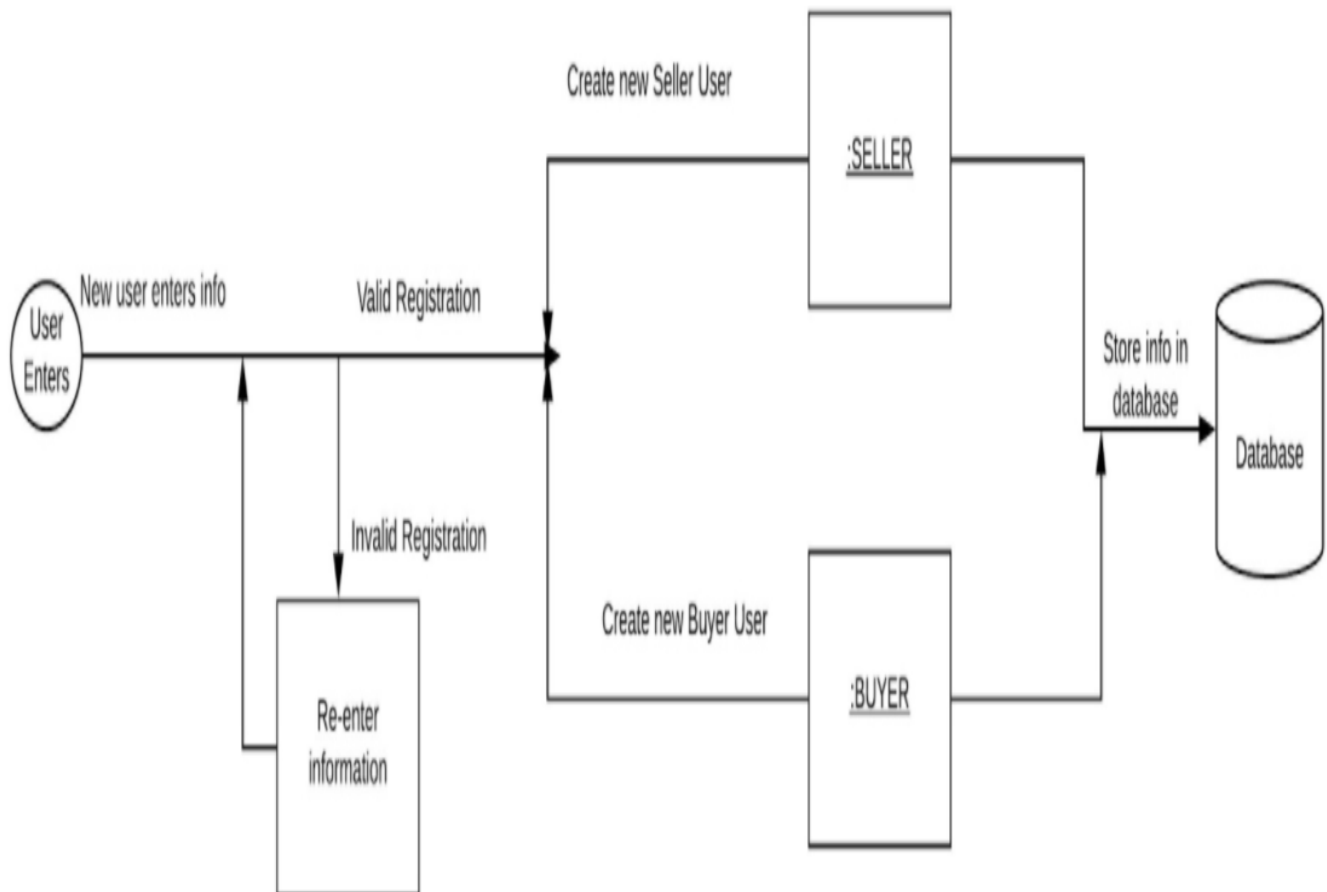
## 6.0 Behavioral Design

In the following section you will find a collaboration model diagram to represent each of the use cases in our use case diagram. This diagram will help us visualize the sequence of steps and what classes will be use in that certain process.

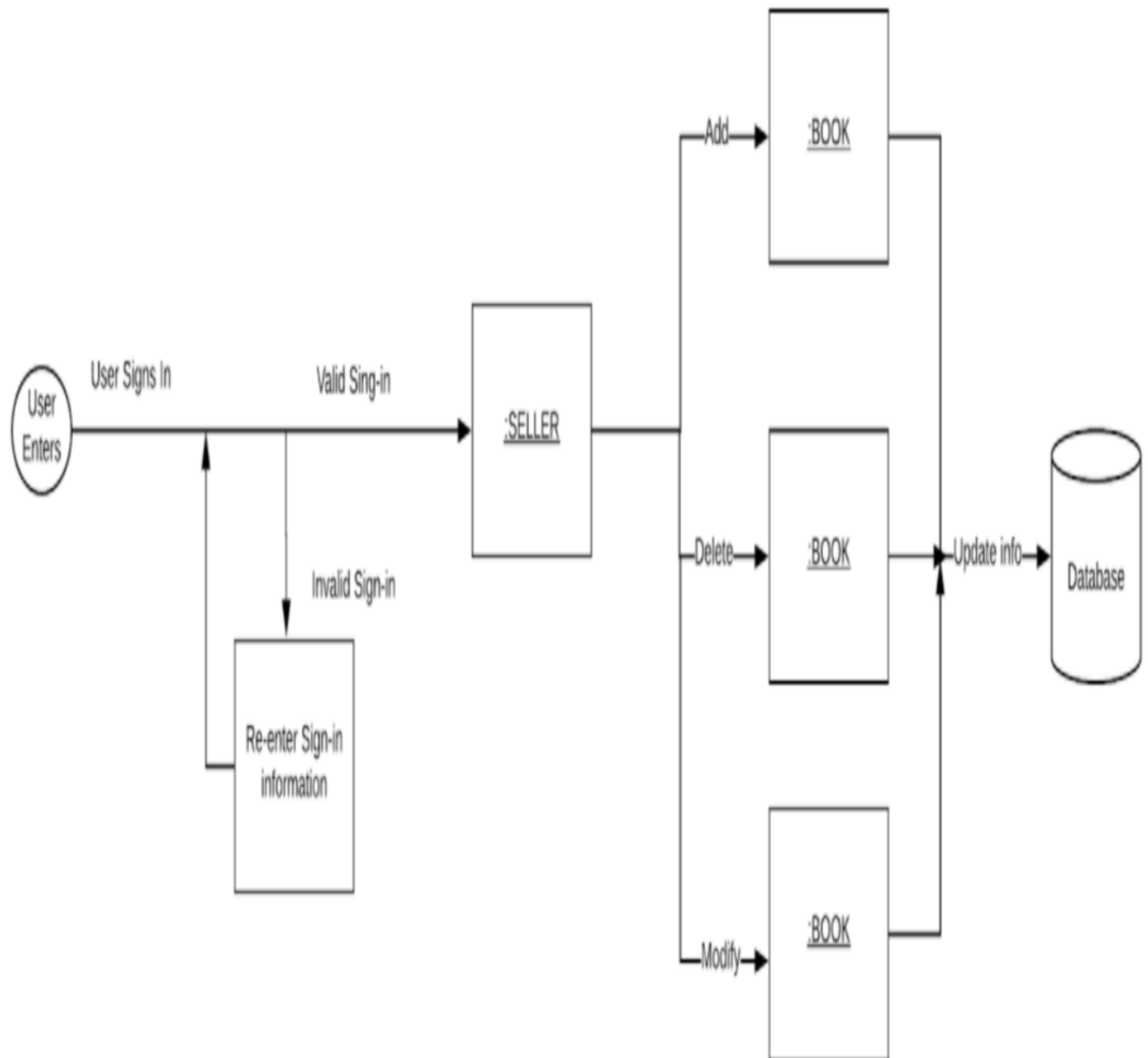
### 6.1 System Services Implementation Model

BUYER/SELLER Registration Case:

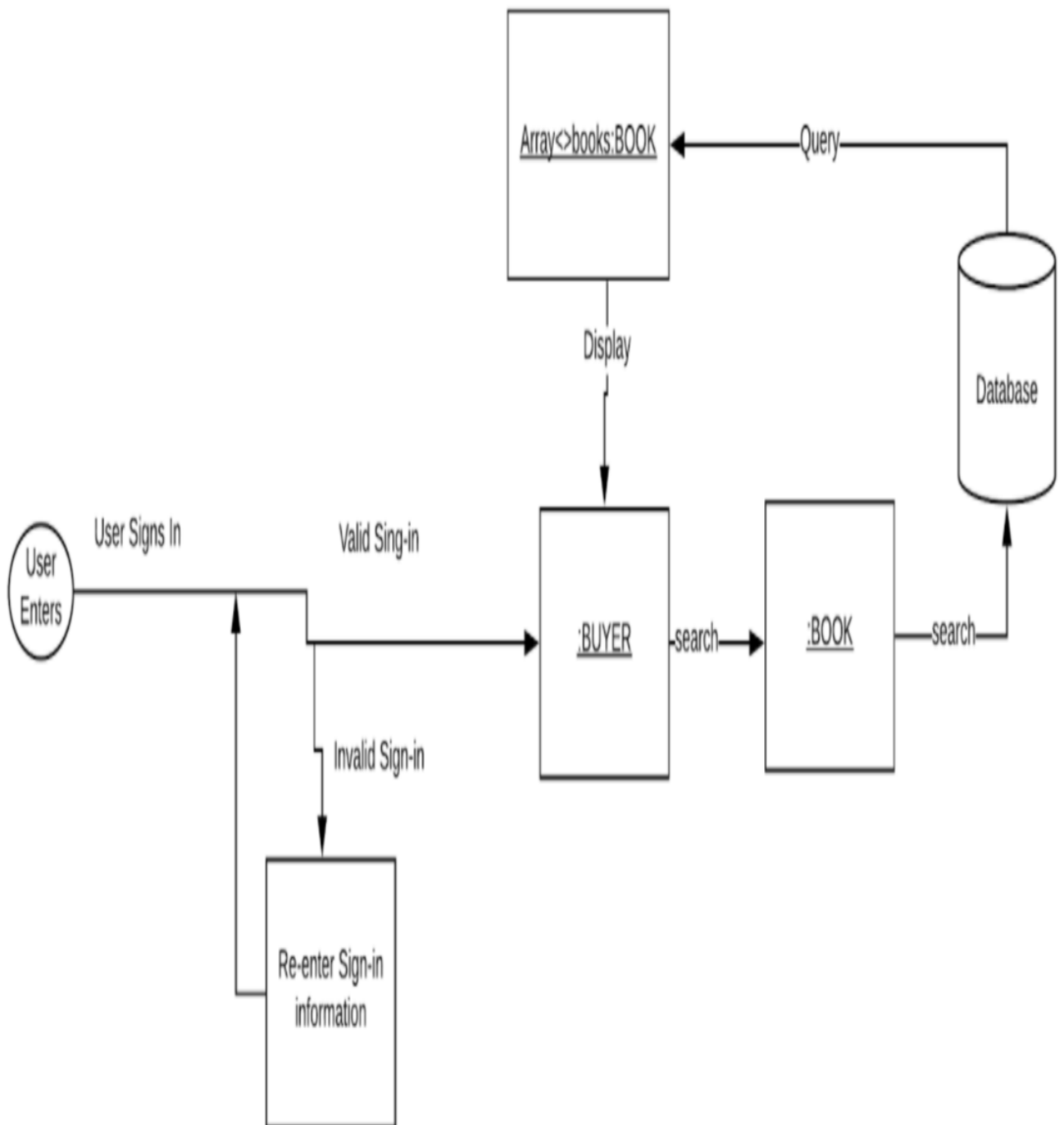
Use Case #1



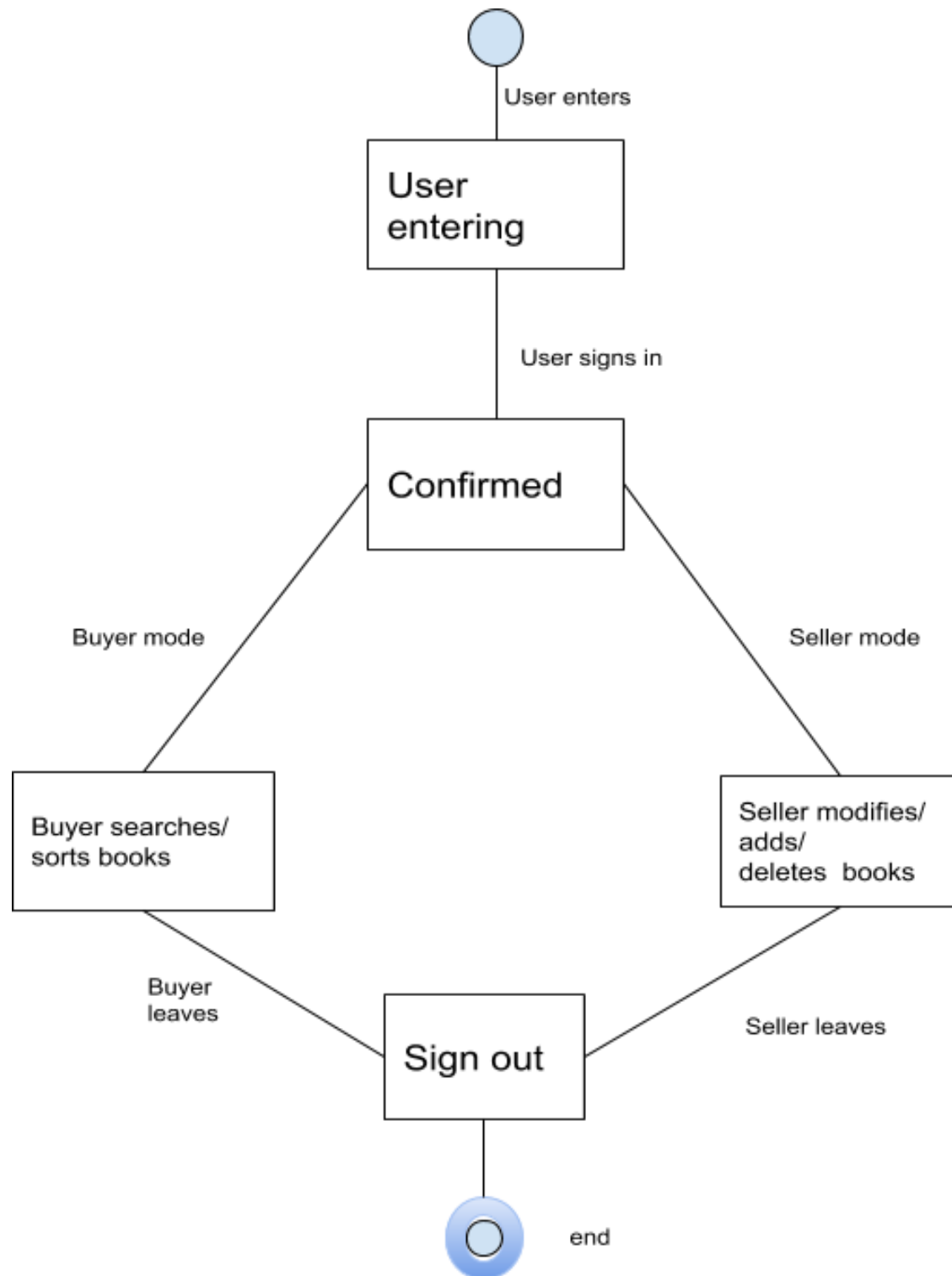
SELLER Add/Delete/Modify BOOK Case:  
Use Case #2,3,4



BUYER Search/Sort BOOK Results Case:  
Use Case #5.6



## 6.2 State Chart Diagram(s)



## 7.0 User Interface Design

---

The following section will describe the basic user interface layout for SPU Book Finder. The first section, will entail any user interface requirements and constraints. The requirements and constraints contain lists of essential attributes the system will have and will not have. The next section will include a window navigation diagram. A window navigation diagram consists of all the interface elements and the relationships between. The last section contains snapshots and descriptions of the main views of SPU Book Finder.

### 7.1 User Interface Requirements and Constraints

---

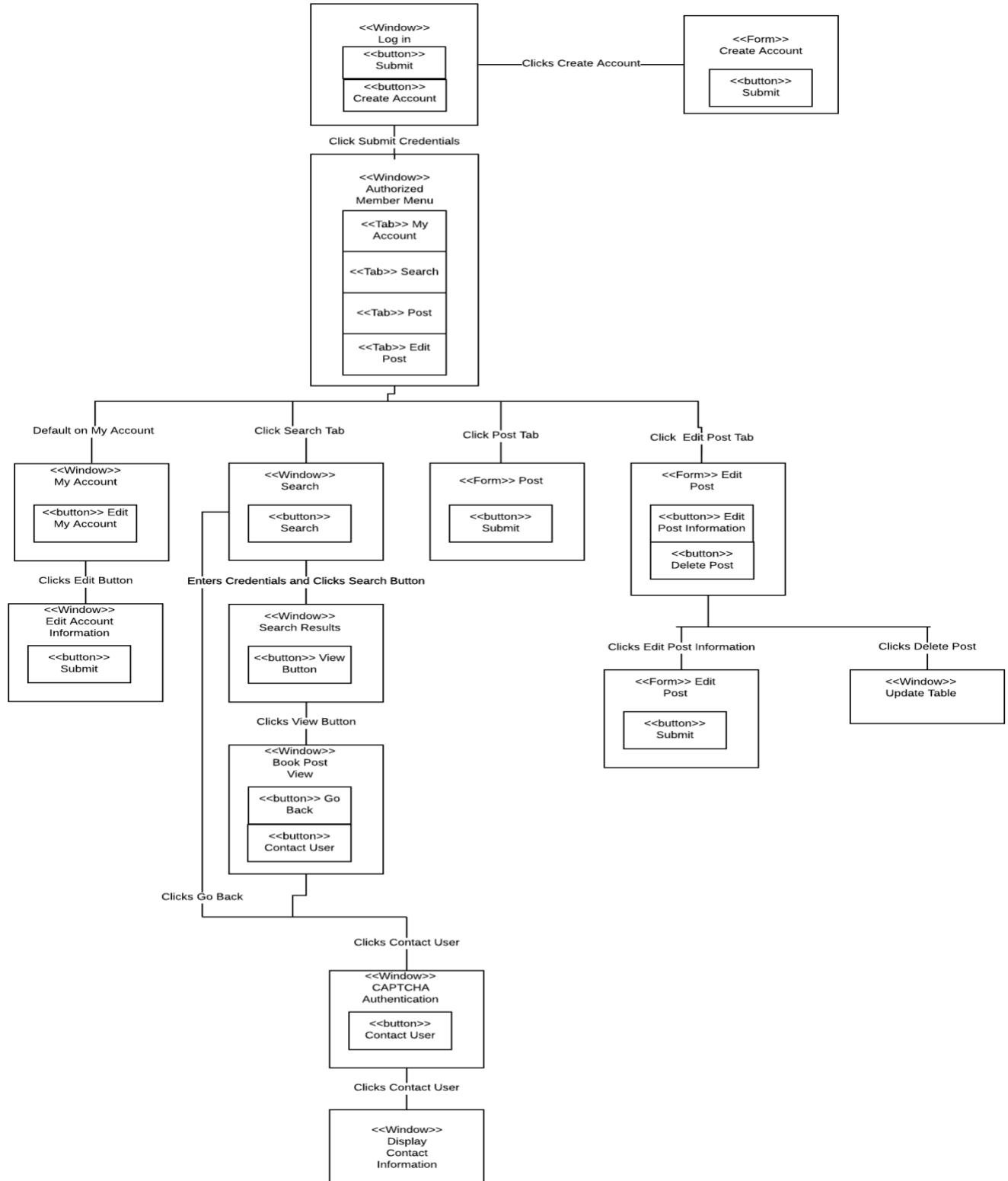
SPU Book Finder, is a web application customized to SPU students. To create a web application that is the best fit for students at SPU, we had to conduct surveys and interviews with our client and other SPU students, to find out what SPU students want versus what they need. The following are the requirements for the user interface based on interviews and surveys conducted by our team:

- Availability
  - The system interface will be available online 27/7 except during routine maintenance.
- Organization
  - The system will have simple tab navigation for the main features of the system (My Account, Search, Post, and Edit).
  - In My Account Window, it will display the information gathered of the user when they first created the account. There will be an Edit Account, that will allow users to edit user account information.
  - In the Search Window, it will allow users to enter textbook credentials. Once the user hits submit, they will get a list of all available textbooks. Users will be able to click on posts to get more information on the textbook and get contact information for the relevant user.
  - In the Post Window, it will prompt users to a form that will allow users to quickly enter the textbook credentials.
  - In the Edit Window, it will prompt users with a list of all their posts. This section allows them to edit their posts and delete posts for any reason.

The following are the constraints of the user interface based on interviews and surveys conducted by our team:

- Portability
  - The system interface will display the appropriate interface based on if the user is on a PC, mobile device, or tablet.
- Security
  - The system interface will incorporate CAPTCHA to ensure security.
  - The system will validate the user email to check if it is an SPU email.
  - The system will allow the user to only fill out private information about themselves that they are comfortable sharing with other users.
  - The system will ask the user to agree to the terms and conditions of sharing their private information they have entered to be shared with others on our system before allowing the user to make an account.

## 7.2 Window Navigation Diagram





### 7.3 Screen Interface Design

SPU Book Finder

spubookfinnder.com

username: Text

password: Text

Submit Create Account

Figure 0. Log in screen. Option of logging in with credentials or creating an account.

SPU Book Finder

spubookfinnder.com

Create Account

Text

Text

Text

Text

Submit

Figure 1. Window prompt if Create Account is pressed. The user must only fill out information that they would want to share with other users, so only the fields of name and email will be required.

The image shows a web browser window titled "SPU Book Finder" with the address bar displaying "spubookfinnder.com". The browser's navigation bar includes back, forward, refresh, and home icons. On the left side of the page, there is a vertical menu with a placeholder icon (a square with an 'X') at the top, followed by the text "My Account". Below this text are four buttons: "Search", "Post", and "Edit", each on its own line. The main content area is titled "My Account Information". It contains four text input fields, each labeled "Text", stacked vertically. At the bottom of this section is a button labeled "Edit Account", which is circled in red.

Figure 2. Window prompt when Submit clicked in Figure 0. This is the first page a user views when logged in, a basic summary of their account.

The image shows a web browser window titled "SPU Book Finder" with the address bar displaying "spubookfinnder.com". The browser's navigation bar includes back, forward, refresh, and home icons. On the left side of the page, there is a vertical menu with a placeholder icon (a square with an 'X') at the top, followed by the text "My Account". Below this text are four buttons: "Search", "Post", and "Edit", each on its own line. The main content area is titled "Edit Account Information". It contains four text input fields, each labeled "Text", stacked vertically. At the bottom of this section is a button labeled "Submit".

Figure 3. Window prompt when Edit Account clicked in Figure 2. This page will allow users to change any account information they want to share with other users. Name and email address fields are required.

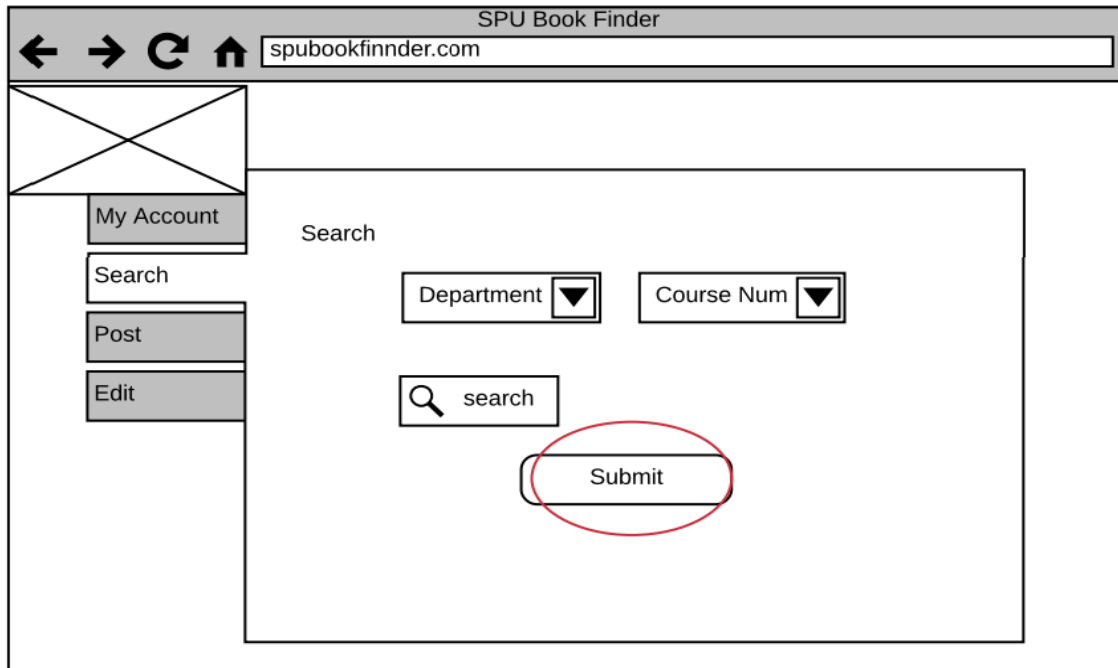


Figure 4. Window prompt when Search Tab clicked. User is able to enter any textbook credentials to begin their search. Some of the options are Department and Course Number, ISBN, Title, etc.

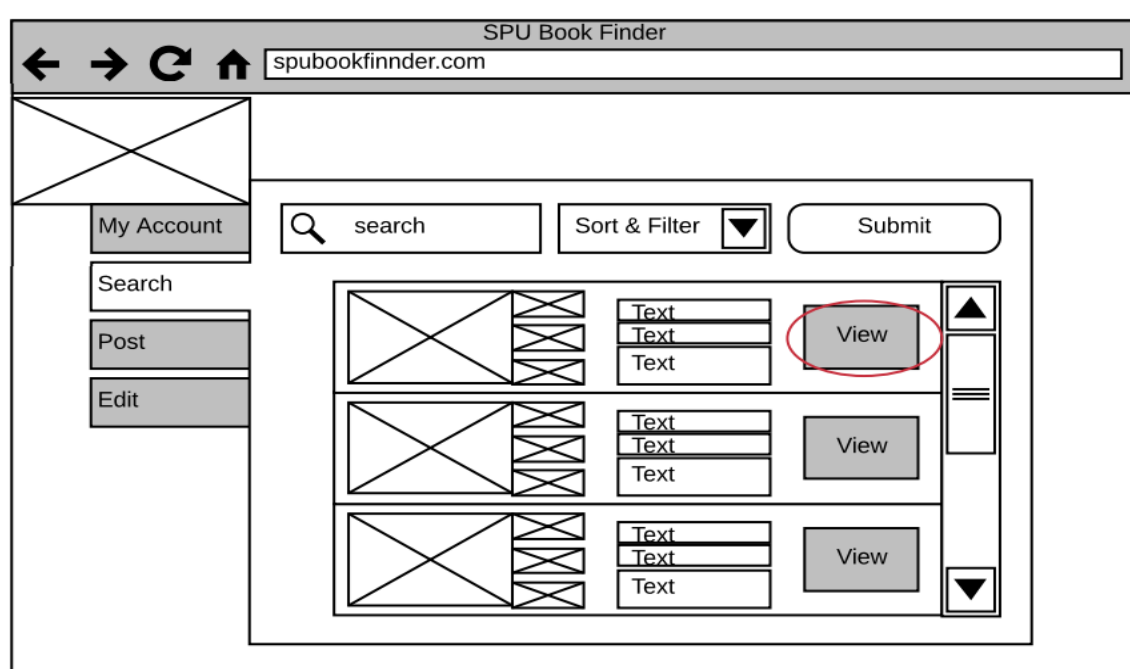


Figure 5. Window prompt when search credentials entered and Submit clicked from Figure 4. A list view will only be displayed if there are relevant books available in our database. Otherwise, it will be empty and allow users to go back.

SPU Book Finder

spubookfinnder.com

My Account

Search

Post

Edit

Text

Text

Text

CAPTCHA

Go Back

Contact User

Figure 6. Window prompt when “View” is clicked in Figure 5. This window contains a detailed description and more information about the current textbook the user is interested in. There are two buttons on this screen. The “Go Back” button will take the user to the list with the relevant textbooks they searched for, which is Figure 5. The “Contact User” action will be described in Figure 7.

SPU Book Finder

spubookfinnder.com

My Account

Search

Post

Edit

Text

Text

Text

User Contact Information

Text

Text

Go Back

Contact User

Figure 7. Window prompt when Contact User clicked and CAPTCHA completed from Figure 6. This will display the user contact information that the user wanted others to contact them on, could be a combination of attributes: name, email, phone number, social media, etc.

SPU Book Finder

spubookfinnder.com

My Account

Search

Post

Edit

Text

Text

Text

☒ Authorize Signature

Submit

Figure 8. Window prompt when Post Tab clicked. The window will display a form with options to enter pictures, title name, ISBN, department and course number, condition, etc, of a textbook they want to post to our system. The user must authorize that they wish to post, validate that the credentials they have entered are correct, and that they wish to display their contact information onto our system. Without the authorization signature, the post will not be able to be displayed on our system but will be kept as a draft.

SPU Book Finder

spubookfinnder.com

My Account

Search

Post

Edit

Book	Draft	Status
CSC 1230	<input type="checkbox"/>	<input checked="" type="checkbox"/>
CSC 2431	<input type="checkbox"/>	<input checked="" type="checkbox"/>
CSC 3310	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Edit

Delete

Figure 9. Window prompt when Edit Tab clicked. This window shows all the posts that a user has as drafts and live on our website. The two operations for this window is to edit a post(Figure 10) or delete a post(Figure 12).

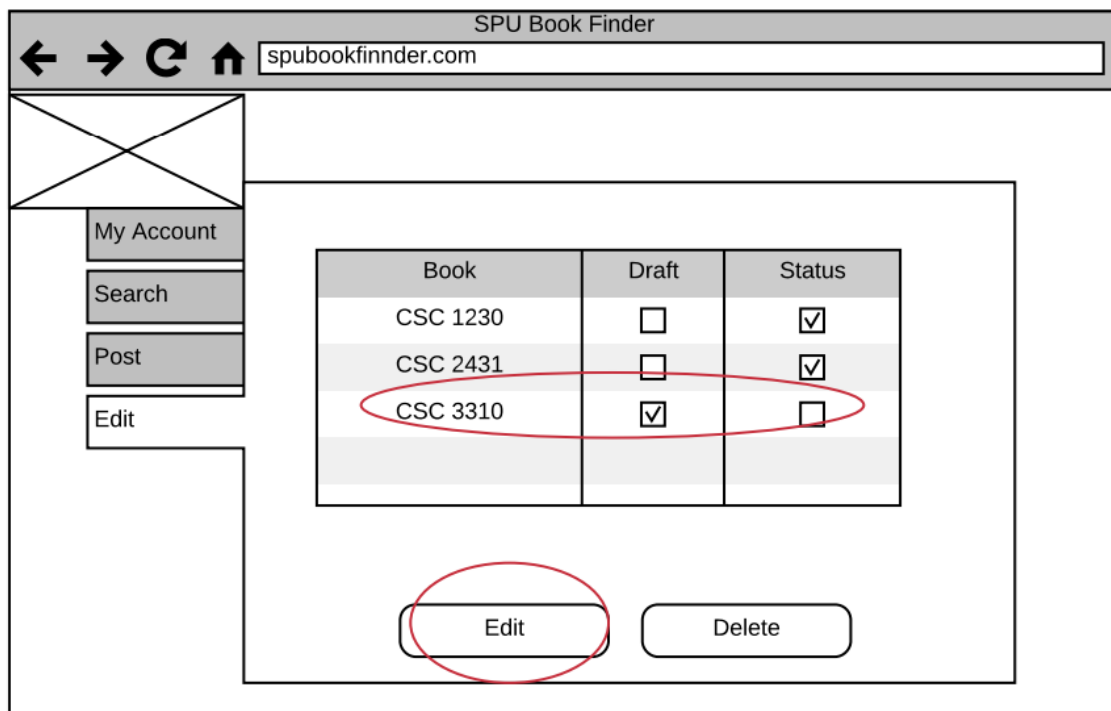


Figure 10. Window prompt when post selected to be edited.

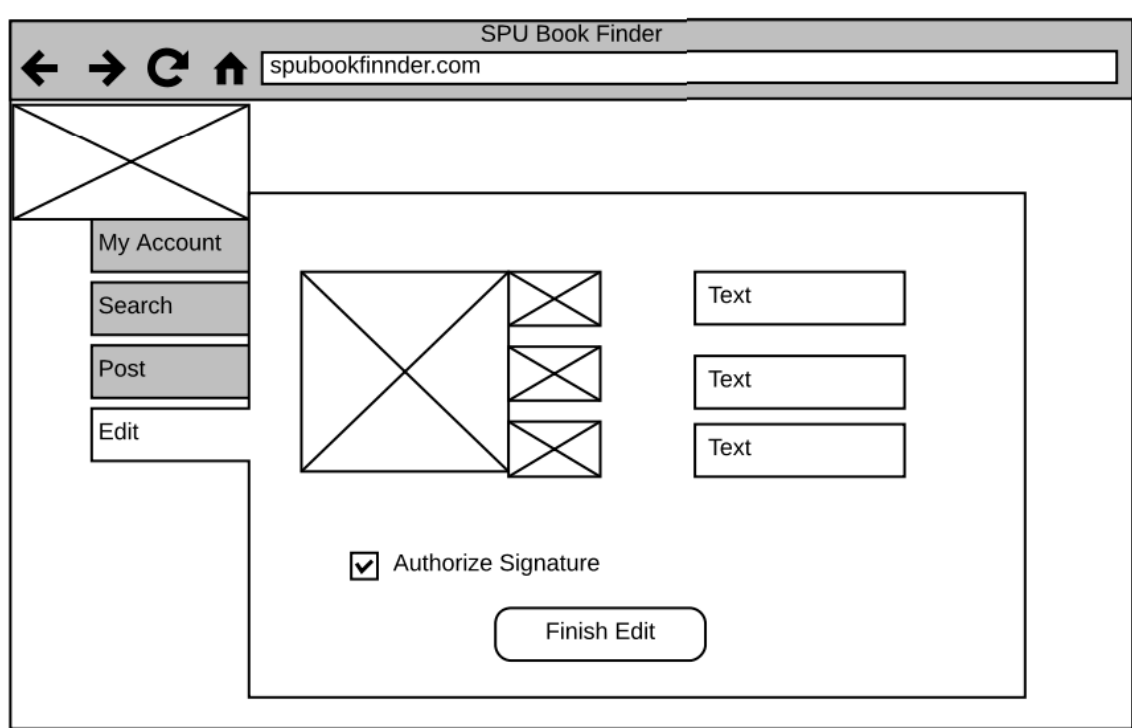


Figure 11. Window prompt after click on Figure 10. The user will be taken to a window similar to Figure 7. It will allow the user to edit any information they have entered or to add any. A new signature validation is required to post onto our system.

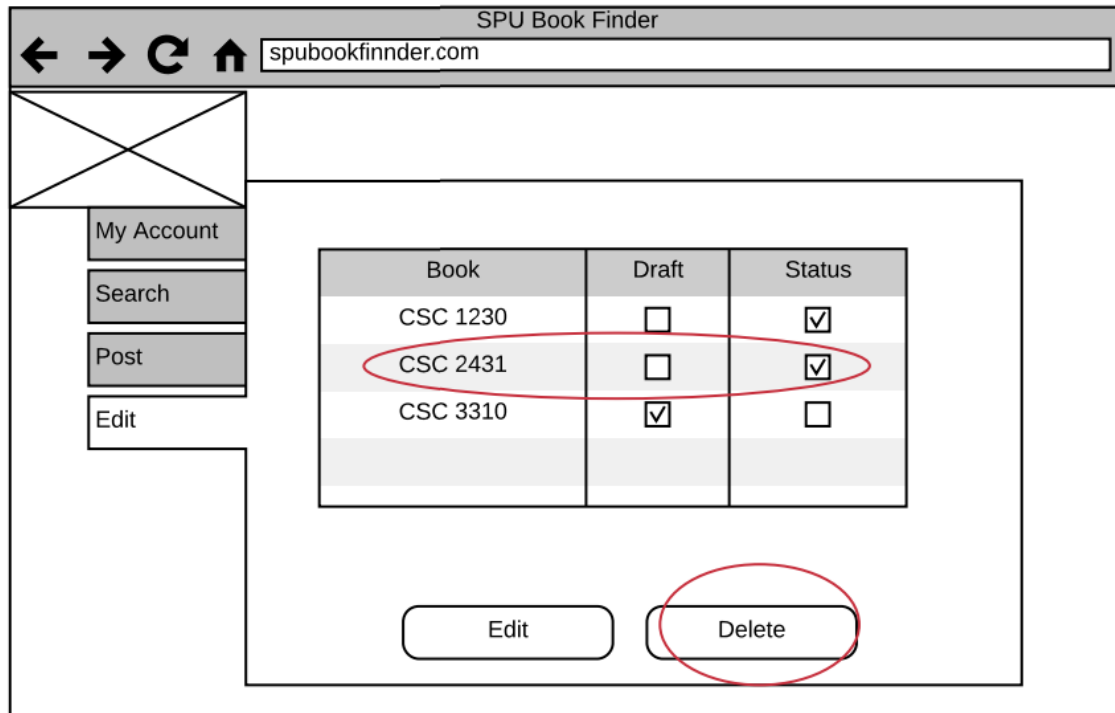


Figure 12. Window prompt when post selected to be deleted. The system will ask the user if they want to delete the post before deleting it, in case the user accidentally clicks delete.

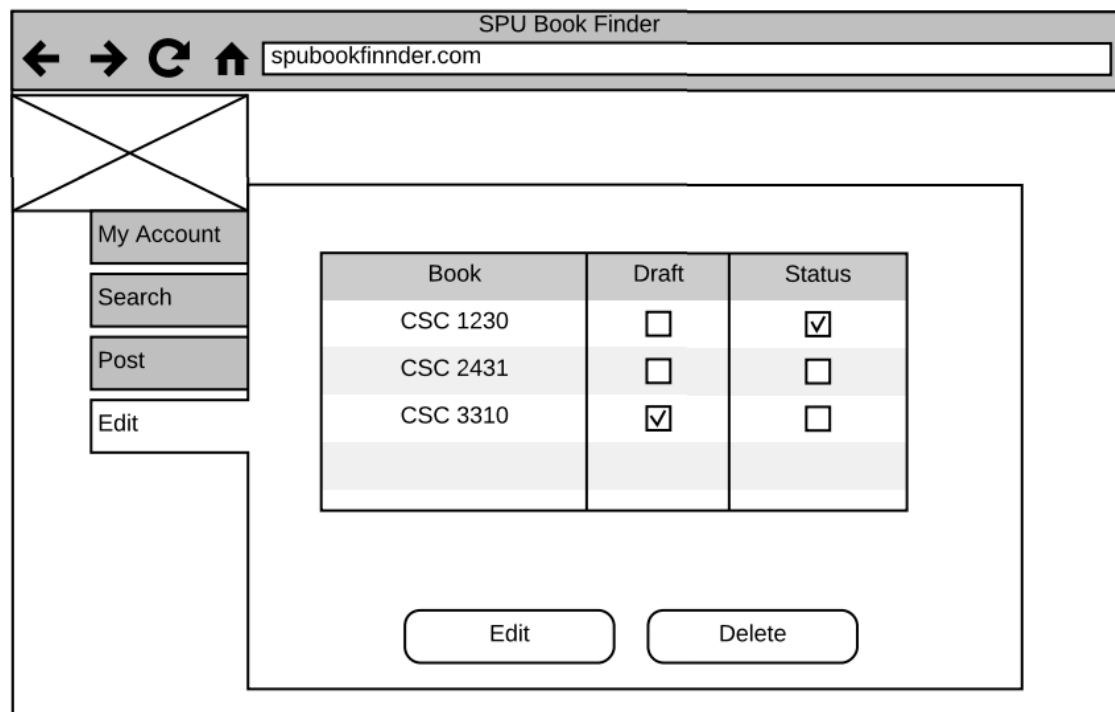


Figure 13. Window prompt after Delete action completed. The post is now erased from our database as well.



## 8.0 Development Plan

---

### 8.1 Development Schedule

---

Week of March 27th

Decide how many pages are needed in total. Design the detailed layout for each page.  
Start working on the general layout of all web pages.

Week of April 2nd

Start working on the HTML/CSS part of the code.

Week of April 9th

Finish all the Design(HTML/CSS) part  
Adding Javascript code to make the website more professional.

Week of April 16th

Finish the Javascript code.

Week of April 23rd

Start working on populating data into the Database.

Week of April 30

Continue populating data into the Database.

Week of May 7th

Start adding PHP code to retrieve and modify data in the database.

Week of May 14th

Finish all the PHP code so that the webpage is responsive to the database.

Week of May 21st

Start testing the program.

Week of May 28

Continue to test the program.

Week June 4th

Finish all the code and testing.