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 Guevara
    a. guevaram@spu.edu
    b. Team Recorder
3. Tianxing Wang
    a. wangt1@spu.edu
4. Cesar Labastida
    a. labastidac@spu.edu

Table of Contents

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.

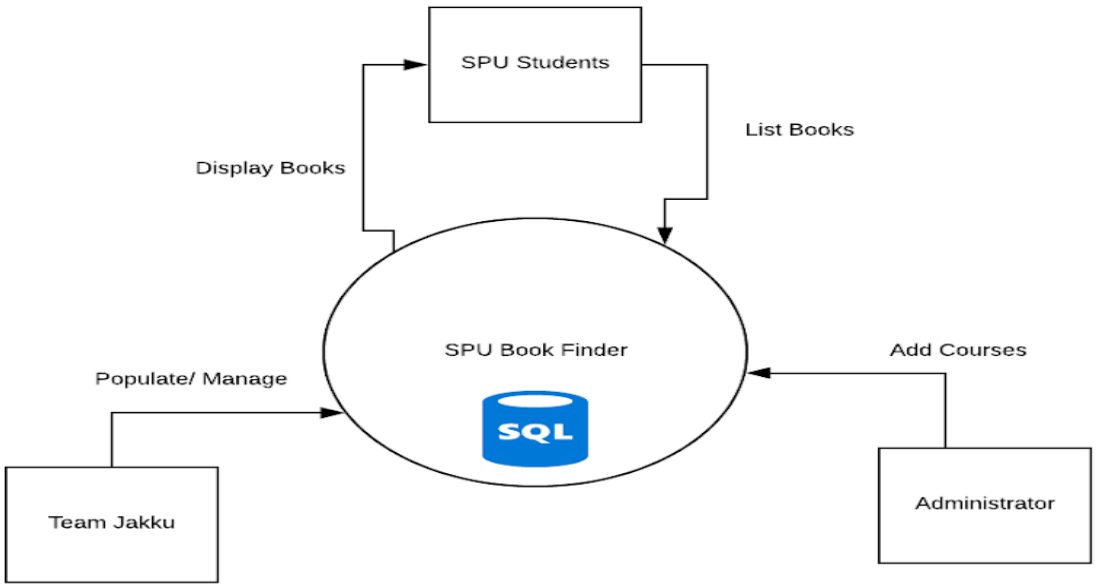
1.2 System Scope

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 keyword, course, or all available in the system.
- The system must display a list of available textbooks to users.
- The system must provide a safe method for a buyer to contact a seller.
- 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, Awardspace, Digital Ocean, or Heroku.
- SPU Book Finder will need to prevent fraudulent and pirated selling of textbooks.
    - Our team will mitigate this by making 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

Create Account	The system must only allow SPU students to create an account
Use-Case(s):	1
Initial Acceptance Tests:	• The user must use their SPU email to create an account • Once an account is created the user must receive an email in which he/she confirms their account • Once the email is confirmed the user can access the SPU Book Finder system

Posting New Book	The system must allow a user to post their book for sale and delete their posting
Use-Case(s):	2, 3

Initial Acceptance Tests:	• Attempt to post with missing information
	  • Name, ISBN Number, or edition missing
	  • Book condition and price missing
	• Attempt to post a book that is not part of the SPU curriculum or database
	• Attempt to delete a post that has been deleted

Change Password	The system must allow users to change their password
Use-Case(s):	4
Initial Acceptance Tests:	• The old password provided does not match their current password • The new password provided matches their old password • The new password provided and the confirm new password do not match

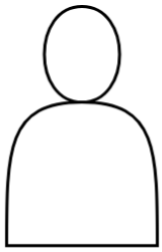
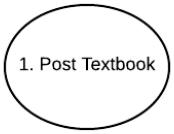
Textbook Search	The system must allow users to search for textbooks by providing a keyword, selecting a course from a drop-down menu, or displaying all available books
Use-Case(s):	5
Initial Acceptance Tests:	• Keyword entered does not return a match • Keyword entered returns more than one match • Selected SPU course does not return a match • Selected SPU course returns more than one match • User searches for a book posting that has been deleted

Communication	The system must allow buyers to communicate with sellers through a message page or message button in the search results of the interested book
Use-Case(s):	6
Initial Acceptance Tests:	• The user provides the incorrect username to message • The person messaged should receive an email notification

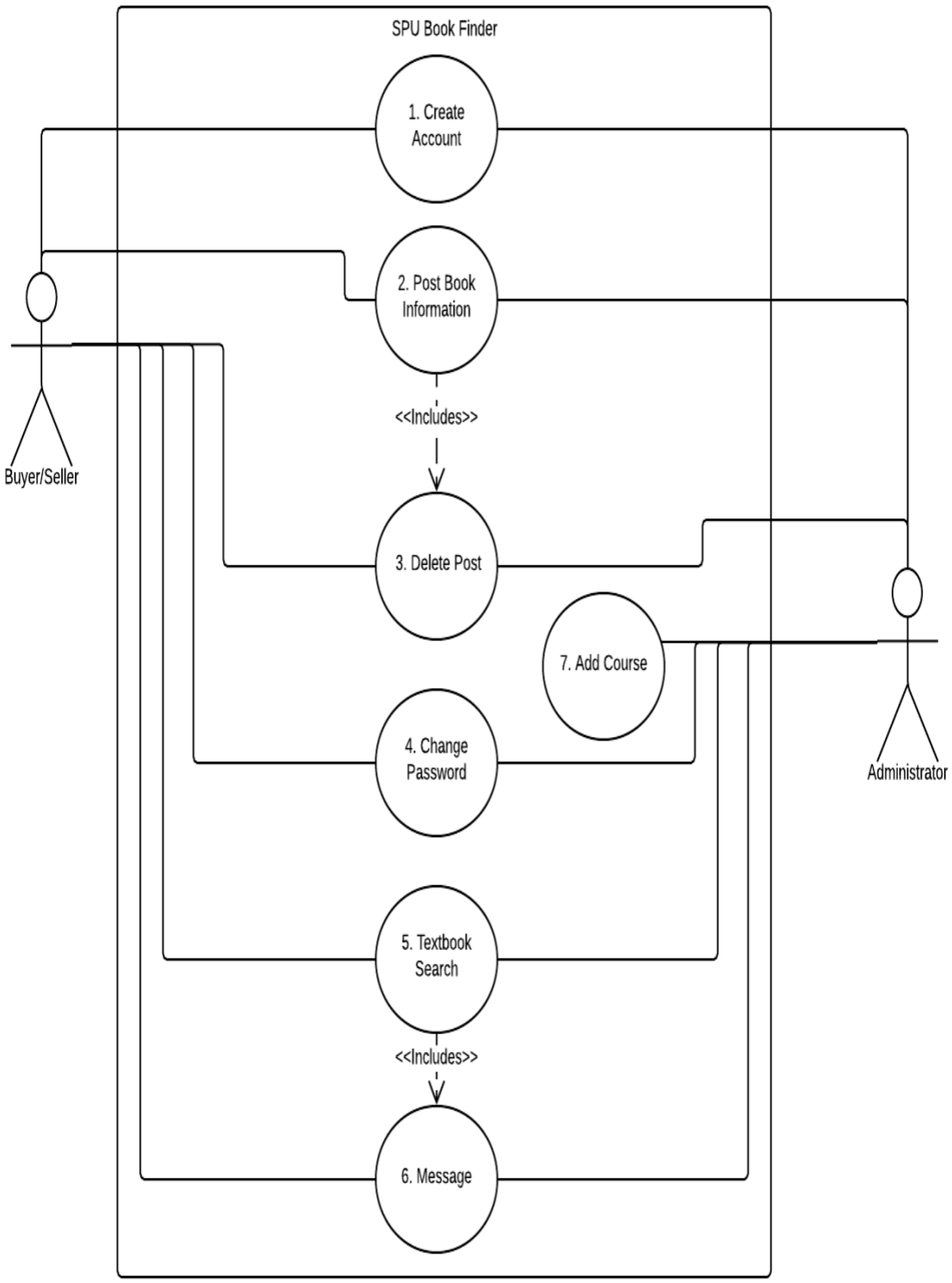
Add Course	The system must allow the database administrator to add a new course to the database
Use-Case(s):	7
Initial Acceptance Tests:	• The added course already exists

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

Use-Case name: Create Account	**ID**: 1	**Importance**: High
Primary actor: Buyer, Seller, and Administrator	**Use-Case type**: Detailed, Essential	
Stakeholders and interests: Buyer, Seller, and Administrator wants a secure way to access the system		
Brief description: 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		
Trigger: User fills out "Sign up" section **Type** (circle one): External		
Relationships: **Association**: Buyer, Seller, and Administrator **Include**: - **Extend**: - **Generalization**: -		
Normal flow of events: 1. User selects "Sign up" section 2. The user enters their information such as SPU email, username 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		
Subflows: None		
Alternate / exceptional flows: 4a: If the user attempts to access the system before confirming their email, access to the system should be denied		

Use-Case name: Post Book Information	**ID**: 2	**Importance**: High
Primary actor: Seller	**Use-Case type**: Detailed, Essential	
Stakeholders and interests: Seller: wants to sell their books and Buyer: wants to buy used textbooks		
Brief description: This use case describes the process of posting a book to the SPU Book Finder system.		
Trigger: Seller selects "Post" **Type** (circle one): External		
Relationships: **Association**: Seller **Include**: 4. Delete post **Extend**: - **Generalization**: -		
Normal flow of events: 1. Seller selects "Post" from the side navigation menu 2. Seller will be taken to a page asking them to fill out information about their book. 3. They have two options to post a book a. Select a course and book from drop down menus i. Then fill out other relevant information about the book b. Select a course from drop down menu i. Then fill out ISBN, book title, author, and other relevant information about the book 4. Select "Submit" and the post will now become public and stored in the database		
Subflows: None		
Alternate / exceptional flows: 4a: After a book is sold, perform use case 3. Delete Post		

Use-Case name: Delete Post	**ID**: 3	**Importance**: High
Primary actor: Seller	**Use-Case type**: Detailed, Essential	
Stakeholders and interests: 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		
Brief description: This use-case describes the process of deleting a post from SPU Book Finder.		
Trigger: The user selects "Delete" **Type** (circle one): External		
Relationships: **Association**: Seller **Include**: - **Extend**: - **Generalization**: -		
Normal flow of events: 1. The user selects "Delete" 2. The user selects the book they want to delete from a drop down menu 3. The available book database is now updated		
Subflows: None		
Alternate / exceptional flows: None		

Use-Case name: Change Password	**ID**: 4	**Importance**: High
Primary actor: Buyer, Seller, and Administrator	**Use-Case type**: Detailed, Essential	
Stakeholders and interests: Buyer, Seller, and Administrator wants to be able to change their password.		
Brief description: This use-case describes the process of a user changing their password		
Trigger: The user selects "My Account" **Type** (circle one): External		
Relationships: **Association**: Buyer, Seller, and Administrator **Include**: - **Extend**: - **Generalization**: -		
Normal flow of events: 1. The user selects "My Account" 2. The user then provides their old password and their new password and then they confirm their new password. 3. By clicking "Save" the user's account is now updated		
Subflows: None		
Alternate / exceptional flows: None		

Use-Case name: Textbook Search	**ID**: 5	**Importance**: High
Primary actor: Buyer	**Use-Case type**: Detailed, Essential	
Stakeholders and interests: Buyer: wants an efficient way to search for available books Seller: wants to be noticed or found by the buyer		
Brief description: This use case describes the process of searching SPU Book Finder for available books		
Trigger: Buyer selects "Search" from side navigation bar **Type** (circle one): External		
Relationships: **Association**: Buyer **Include**: 6. Message **Extend**: - **Generalization**: -		
Normal flow of events: 1. User selects the "Search" option from the side navigation bar 2. User enters a keyword to search for 3. The user selects a course from drop-down menu 4. The user selects to view all available textbooks 5. The user can then select the book they are interested in to learn more about it or message seller		
Subflows: None		
Alternate / exceptional flows: 5a: If a user selects a message a user from the search results. Perform use-case 6. Message		

Use-Case name: Message	**ID**: 6	**Importance**: High
Primary actor: Buyer and Seller	**Use-Case type**: Detailed, Essential	
Stakeholders and interests: Buyer: wants a way to communicate with the seller Seller: wants a secure way to be contacted by the buyer		
Brief description: This use-case describes the process of a buyer contacting a seller through email		
Trigger: Buyer selects "Message" from the side navigation bar or from the search results "Message" button **Type** (circle one): External		
Relationships: **Association**: Buyer **Include**: - **Extend**: - **Generalization**: -		
Normal flow of events: 1. The user selects "Message" from either the side navigation bar a. Provide the username they would like to message, subject, message, and contact information 2. The user selects "Message" from the button on the search results screen a. The username is automatically filled out, the user just needs to provide the subject, message, and contact information. 3. The Seller receives an email notifying them that a message has been sent to them through the website 4. The Buyer and Seller can now communicate outside of the website		
Subflows: None		
Alternate / exceptional flows: None		

Use-Case name: Add Course	**ID**: 7	**Importance**: High
Primary actor: Administrator	**Use-Case type**: Detailed, Essential	
Stakeholders and interests: Administrator wants to add new course to the database in order for users to be able to post accurate descriptions for their textbooks		
Brief description: This use-case describes the process of adding a course to the SPU Book Finder database		
Trigger: The Administrator selects "Add Course" from the side navigation menu **Type** (circle one): External		
Relationships: **Association**: Administrator **Include**: - **Extend**: - **Generalization**: -		
Normal flow of events: 1. The Administrator selects "Add Course" 2. The Administrator can now provide the Course Department Number and the Course Name 3. The Administrator then selects "Submit" and this course is added to the database		
Subflows: None		
Alternate / exceptional flows: 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

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. The system should be crowdsourced in the future (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 has to provide the book's title or ISBN.
2. The system must provide an organized user interface to ensure use of system
3. The system must provide messages to notify users of successes or failure that happen.
4. The system must be consistent across different platforms
5. The system must have a connection to the internet

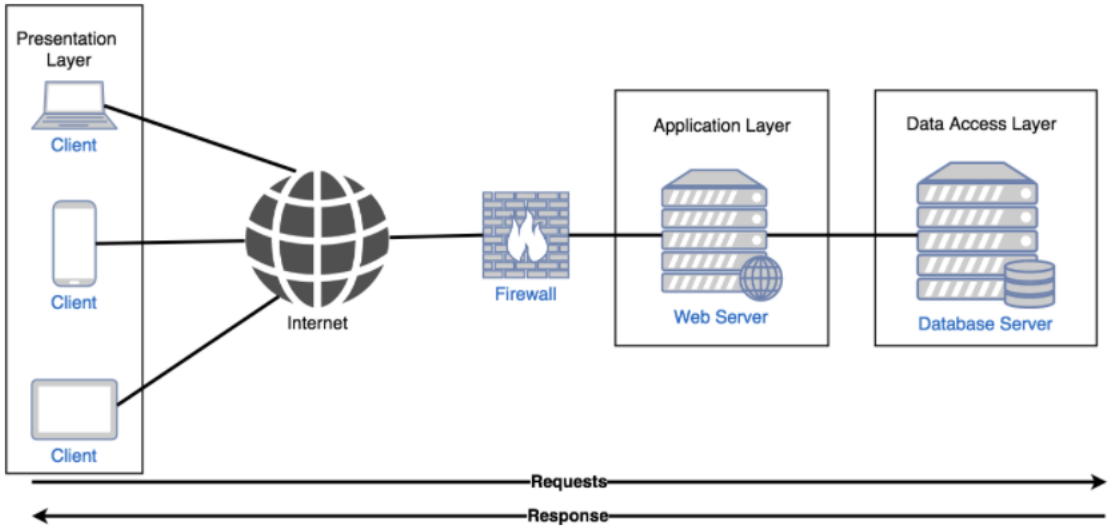
4.0 Development and Operating Environments

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 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

SPU Book Finder will only be accessible to authorized users, because the current version is open to the students at Seattle Pacific University only. Thus, authenticating 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.

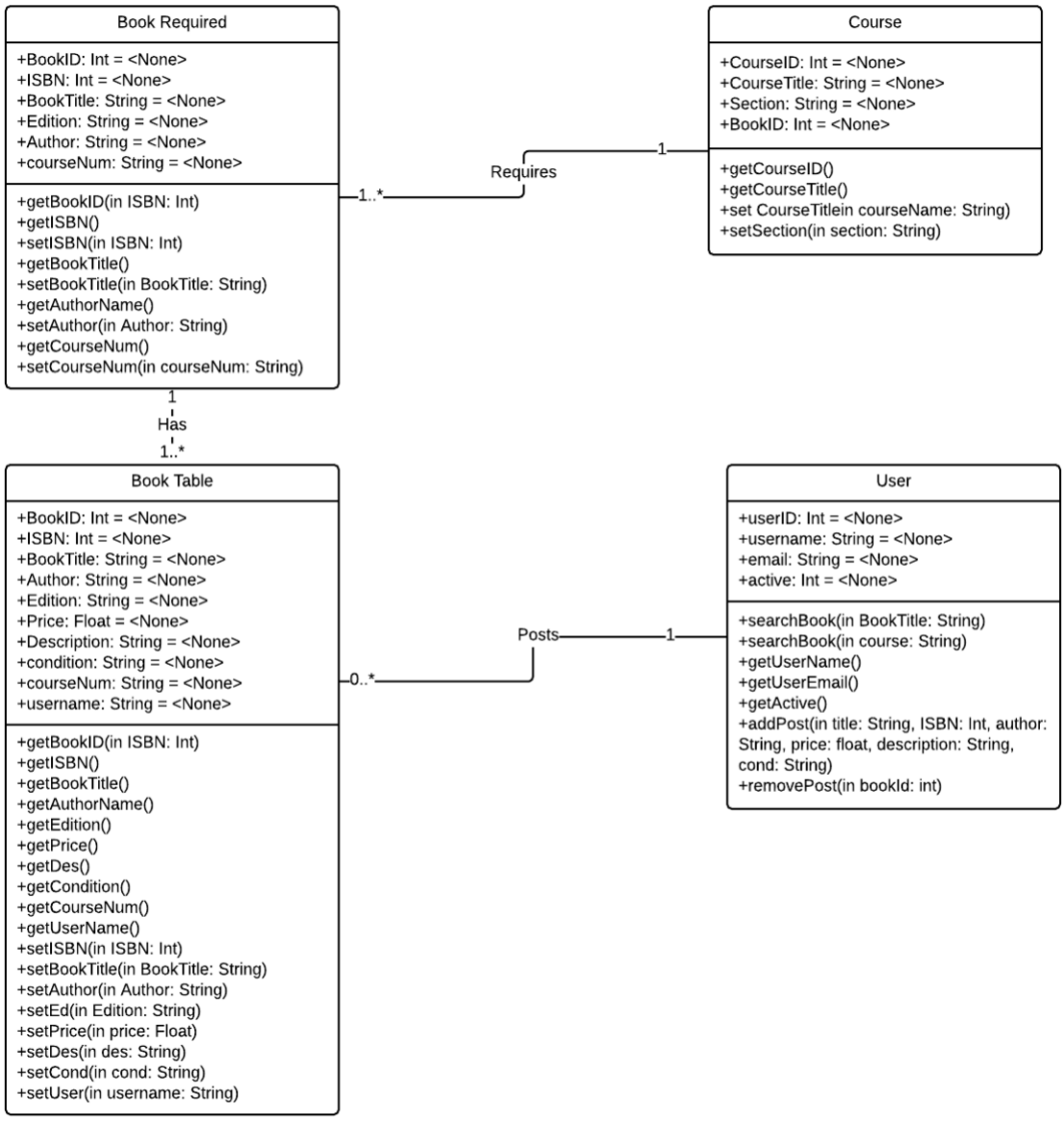
4.5 Development Environment

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. In addition, AwardSpace is used as the hosting provider. PHPMyAdmin is also used as the platform to create and update the database.

5.0 Data(Structural) Design

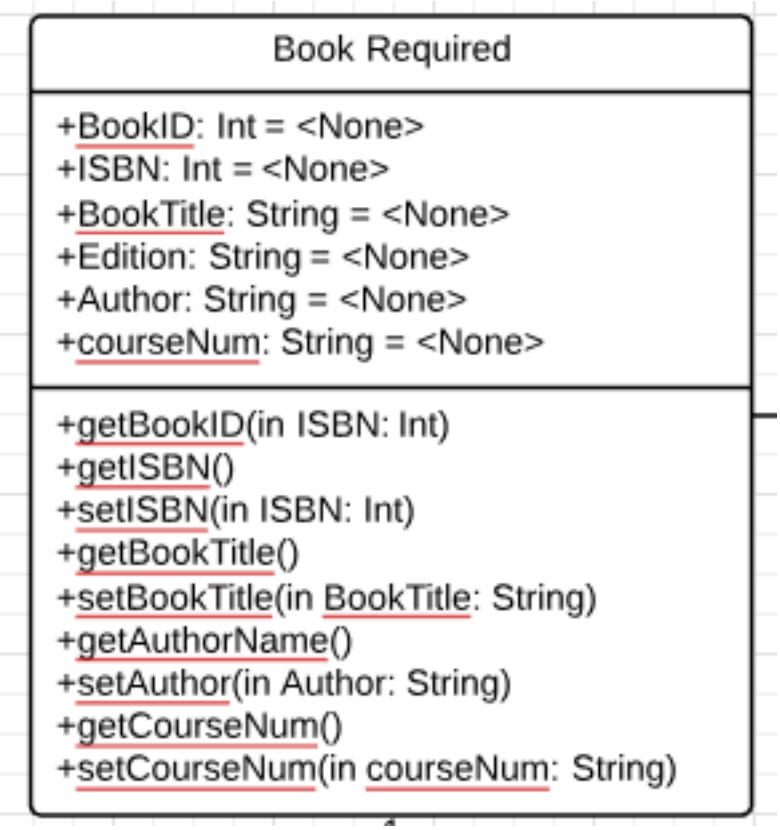
A class diagram is shown to expose the details of SPU Book Finder system. A class diagram is useful to understand our systems functionality and data requirements. Also, this helps us better understand our system's inner dynamics. Additionally, we have included a Data Dictionary that will help understand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

5.1 Class Diagram



5.2 Data Dictionary (Metadata)

BOOK REQUIRED CLASS



Name: Book Required

Visibility: Public

Documentation: This class is in charge of representing each required book in the system. This class will be used for identifying which book posts correspond to the required books.

Attributes: BookID, ISBNNum, BookTitle, Edition, Author, courseNum.
BookID: ID to identify books in the database
ISBNNum: ISBN Number of the book.
BookTitle: Title of the book.
Edition: Edition of the book
Author: Author or Authors of the book, this can be more than one.
courseNum: The course number of the course using the book

Operations:
getBookID(in ISBN: Int)
getISBN()
setISBN(in ISBN: Int)
getBookTitle()

setBookTitle(in BookTitle: String)
getAuthorName()
setAuthor(in Author: String)

ATTRIBUTES

Name	Type	Visibility	Multiplicity	Initial Value
BookID	integer	Global	1	NULL
ISBNum	integer	Global	1	NULL
BookTitle	String	Global	1	NULL
Author	String	Global	1…*	NULL
courseNum	String	Global	1	NULL

OPERATIONS

getBookID();

	Name	Type	Visibility	isQuery	Default
Return	BookID	integer	global	true	0
Parameters	none	none	none	none	none

getISBN();

	Name	Type	Visibility	isQuery	Default
Return	ISBNum	integer	global	true	0
Parameters	none	none	none	none	none

setISBN(int ISBN);

	Name	Type	Visibility	isQuery	Default
Return	none	none	none	none	none
Parameters	ISBNum	integer	global	true	0

getBookTitle();

	Name	**Type**	**Visibility**	**isQuery**	**Default**
Return	BookTitle	String	global	true	Null
Parameters	none	none	none	none	none

setBookTitle(String BookTitle);

	Name	**Type**	**Visibility**	**isQuery**	**Default**
Return	none	none	none	none	none
Parameters	bookTitle	String	global	true	Null

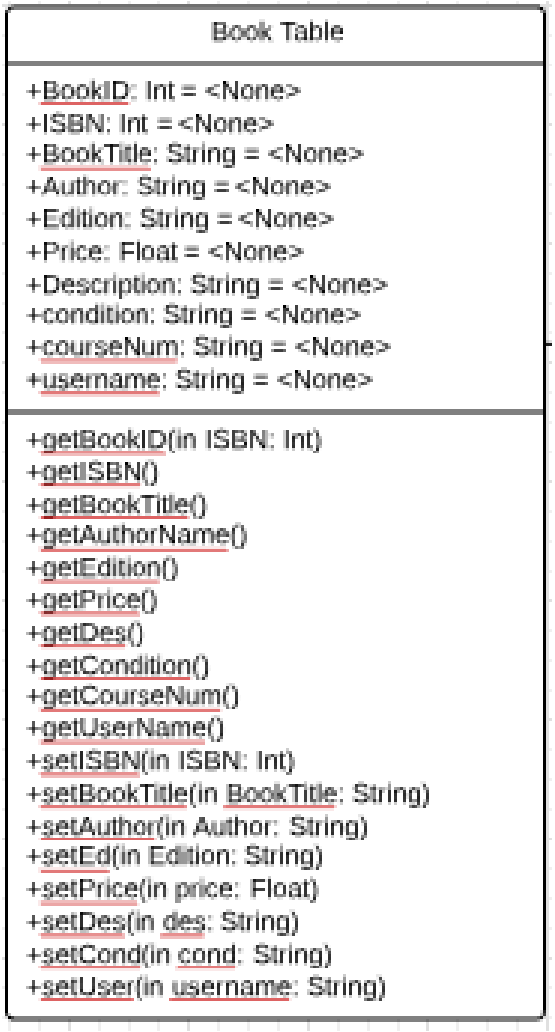
getAuthorName();

	Name	**Type**	**Visibility**	**isQuery**	**Default**
Return	Author	String	global	true	Null
Parameters	none	none	none	none	none

setAuthorName(String authorName);

	Name	**Type**	**Visibility**	**isQuery**	**Default**
Return	none	none	none	none	none
Parameters	authorName	String	global	true	Null

BOOK TABLE CLASS



Name: Book Table

Visibility: Public

Documentation: This class is in charge of representing each book post. This class will be used to get and store information about the book posts.

Attributes: BookID, ISBN, BookTitle, Author, Edition, Price, Description, condition, courseNum, username

BookID: ID to identify books in the database
ISBN: ISBN Number of the book.
BookTitle: Title of the book.
Author: Author or Authors of the book, this can be more than one.
Edition: Edition of the book
Price: Price of the book
Description: Description of book
Condition: Condition of book

courseNum: Course number
Username: User's username

Operations:
getBookID(in ISBN: Int)
getISBN()
getBookTitle()
getAuthorName()
getEdition()
getPrice()
getDes()
getCondition()
getCourseNum()
getUserName()
setISBN(in ISBN: Int)
setBookTitle(in BookTitle: String)
setAuthor(in Author: String)
setEd(in Edition: String)
setPrice(in price: Float)
setDes(in des: String)
setCond(in cond: String)
setUser(in username: String**)**

ATTRIBUTES

Name	Type	Visibility	Multiplicity	Initial Value
BookID	integer	Global	1	NULL
ISBN	integer	Global	1	NULL
BookTitle	String	Global	1	NULL
Author	String	Global	1…*	NULL
Edition	String	Global	1	NULL
Price	Float	Global	1	NULL
Description	String	Global	1	NULL
condition	String	Global	1	NULL
courseNum	String	Global	1	NULL
username	String	Global	1	NULL

OPERATIONS

getBookID();

	Name	**Type**	**Visibility**	**isQuery**	**Default**
Return	BookID	integer	global	true	0
Parameters	none	none	none	none	none

getISBN();

	Name	**Type**	**Visibility**	**isQuery**	**Default**
Return	ISBNum	integer	global	true	0
Parameters	none	none	none	none	none

getBookTitle();

	Name	**Type**	**Visibility**	**isQuery**	**Default**
Return	BookTitle	String	global	true	Null
Parameters	none	none	none	none	none

getAuthorName();

	Name	**Type**	**Visibility**	**isQuery**	**Default**
Return	Author	String	global	true	Null
Parameters	none	none	none	none	none

getEdition();

	Name	**Type**	**Visibility**	**isQuery**	**Default**
Return	Edition	String	global	true	Null
Parameters	none	none	none	none	none

getPrice();

	Name	**Type**	**Visibility**	**isQuery**	**Default**
Return	price	float	global	true	0.0
Parameters	none	none	none	none	none

getDescription();

	Name	**Type**	**Visibility**	**isQuery**	**Default**
Return	description	String	global	true	Null
Parameters	none	none	none	none	none

getCondition();

	Name	**Type**	**Visibility**	**isQuery**	**Default**
Return	condition	String	global	true	Null
Parameters	none	none	none	none	none

getCourseNum();

	Name	**Type**	**Visibility**	**isQuery**	**Default**
Return	courseNum	String	global	true	Null
Parameters	none	none	none	none	none

getUserName();

	Name	**Type**	**Visibility**	**isQuery**	**Default**
Return	username	String	global	true	Null
Parameters	none	none	none	none	none

setISBN(int ISBN);

	Name	**Type**	**Visibility**	**isQuery**	**Default**
Return	none	none	none	none	none
Parameters	ISBNum	integer	global	true	0

setBookTitle(String BookTitle);

	Name	**Type**	**Visibility**	**isQuery**	**Default**
Return	none	none	none	none	none
Parameters	bookTitle	String	global	true	Null

setAuthor(String authorName);

	Name	**Type**	**Visibility**	**isQuery**	**Default**
Return	none	none	none	none	none
Parameters	authorName	String	global	true	Null

setEd(String edition);

	Name	**Type**	**Visibility**	**isQuery**	**Default**
Return	none	none	none	none	none
Parameters	edition	String	global	true	Null

setPrice(float price);

	Name	**Type**	**Visibility**	**isQuery**	**Default**
Return	none	none	none	none	none
Parameters	price	float	global	true	0.0

setDescription(String description);

	Name	**Type**	**Visibility**	**isQuery**	**Default**
Return	none	none	none	none	none
Parameters	decription	String	global	true	Null

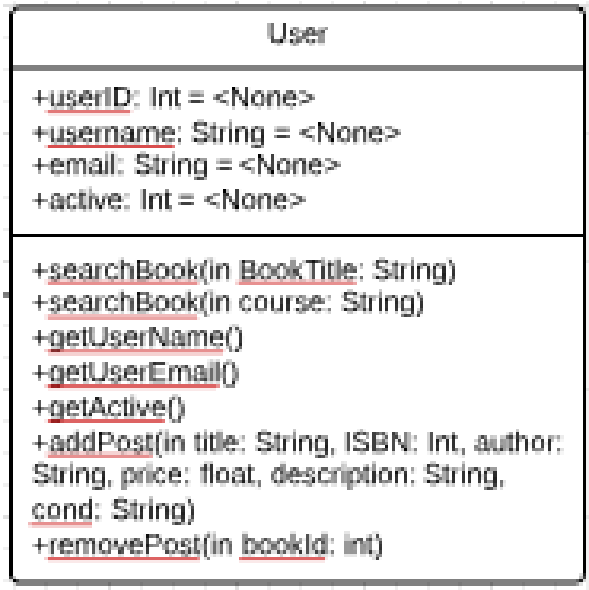
setCond(String condition);

	Name	**Type**	**Visibility**	**isQuery**	**Default**
Return	none	none	none	none	none
Parameters	condition	String	global	true	Null

setUser(String username);

	Name	**Type**	**Visibility**	**isQuery**	**Default**
Return	none	none	none	none	none
Parameters	username	String	global	true	Null

USER CLASS



Name: User

Visibility: Public

Documentation: USER class will be used as a class to store user information such as the user's username, email, and whether their account has been confirmed.

Attributes: userID, username, email, active.
userID: ID to identify users in the database
username: username of the user
email: user's email
Edition: Edition of the book
active: Whether the user's account has been confirmed

Operations:
searchBook(in BookTitle: String)
searchBook(in course: String)
getUserName()
getUserEmail()
getActive()
addPost(in title: String, ISBN: Int, author: String, price: float, description: String, cond: String)
removePost(in bookId: int)

ATTRIBUTES:

Name	Type	Visibility	Multiplicity	Initial Value
userID	Integer	Global	1	NULL
userName	String	Global	1	NULL
email	String	Global	1	NULL
active	Integer	Global	1	0

OPERATIONS:

searchBook(String BookTitle);

	Name	Type	Visibility	isQuery	Default
Return	book	BOOK	global	true	Null
Parameters	title	String	global	true	Null

searchBook(String course);

	Name	Type	Visibility	isQuery	Default
Return	book	BOOK	global	true	Null
Parameters	courseTitle	String	global	true	Null

getUserName()

	Name	Type	Visibility	isQuery	Default
Return	username	String	global	true	Null
Parameters	none	none	none	none	none

getUserEmail()

	Name	Type	Visibility	isQuery	Default
Return	email	String	global	true	Null
Parameters	none	none	none	none	none

getActive()

	Name	Type	Visibility	isQuery	Default
Return	active	integer	global	true	0
Parameters	none	none	none	none	none

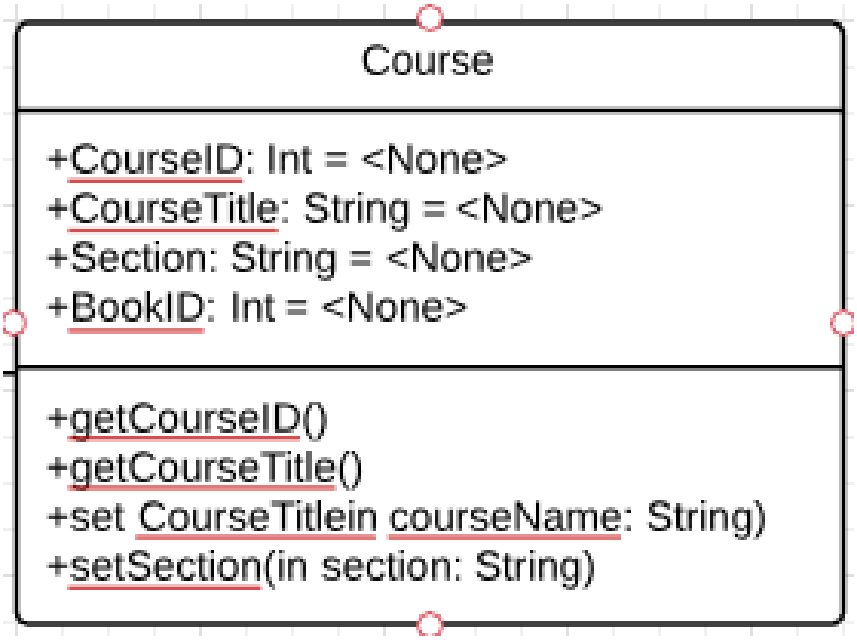
addPost(Book book)

	Name	Type	Visibility	isQuery	Default
Return	success	boolean	global	false	Null
Parameters	book	BOOK	global	true	Null

removePost(BOOK selectedBook)

	Name	Type	Visibility	isQuery	Default
Return	success	boolean	global	false	Null
Parameters	selectedBook	BOOK	global	true	Null

COURSE CLASS



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:
CourseID: Identifies the course
CourseTitle: Title of the course
Section: Course section
BookID: Identifies book

ATTRIBUTES:

NAME	TYPE	Visibility	Multiplicity	Initial Value
courseID	integer	Local	1	NULL
courseTitle	String	Local	1	NULL
Section	String	Local	1	NULL
BookID	integer	Local	1	NULL

OPERATIONS:

getCourseID()

	Name	Type	Visibility	isQuery	Default
Return	courseID	integer	global	true	Null
Parameters	none	none	none	none	none

getCourseTitle

	Name	Type	Visibility	isQuery	Default
Return	courseName	String	global	true	Null
Parameters	none	none	none	none	none

setCourseTitle(String courseName);

	Name	Type	Visibility	isQuery	Default
Return	none	none	none	none	none
Parameters	courseName	String	global	true	Null

setSection(string couseNum);

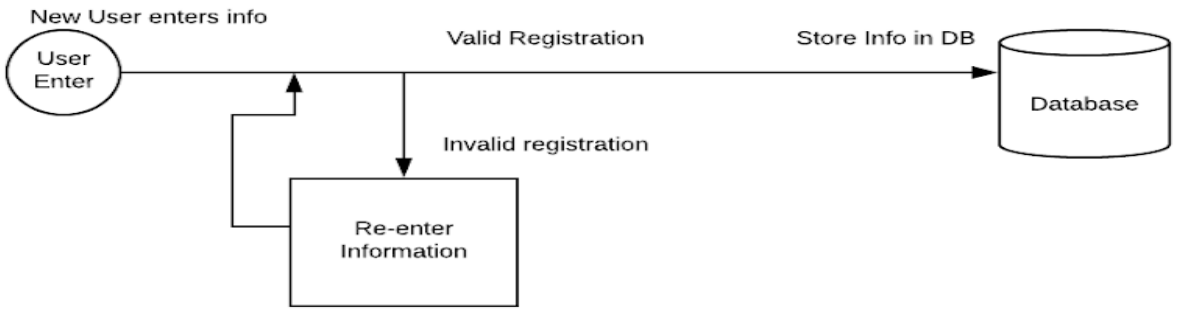
	Name	Type	Visibility	isQuery	Default
Return	none	none	none	none	none
Parameters	courseNum	string	global	true	0

6.0 Behavioral Design

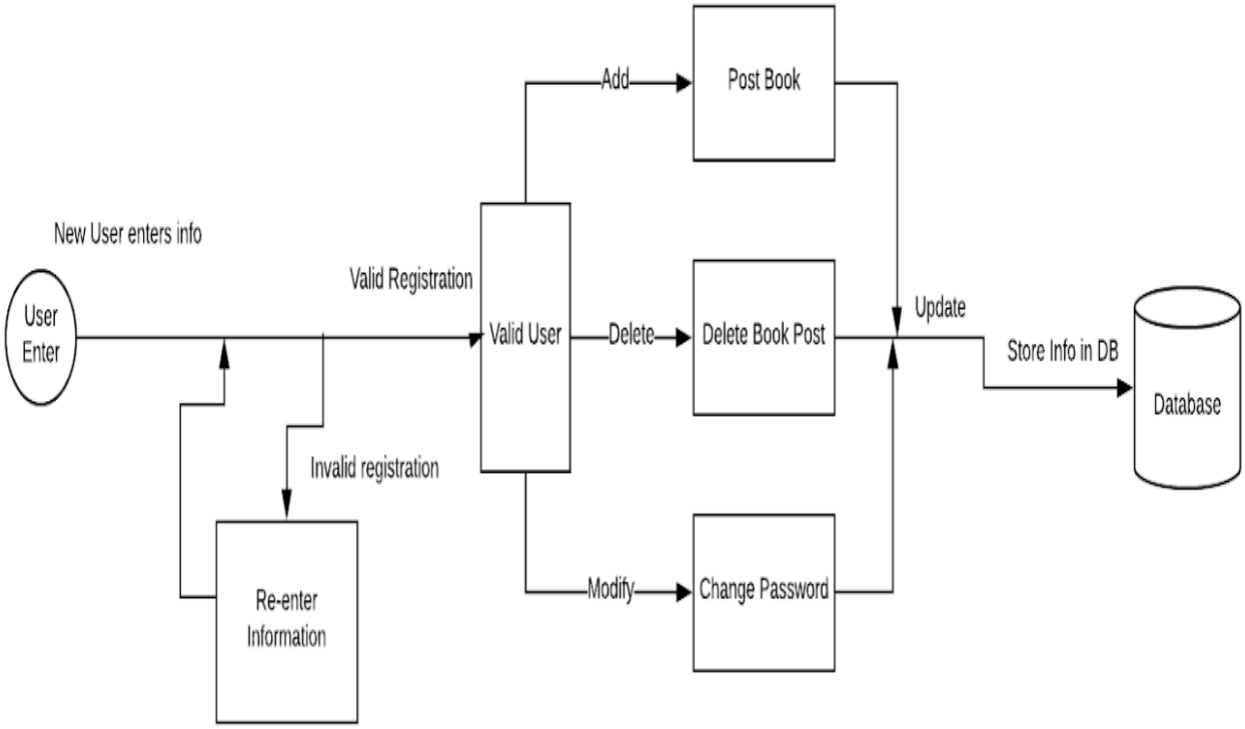
In the following section you will find the system services implementation model diagram to represent each of the use cases in our use case diagram. This diagram will help us visualize the sequence of steps and what classes will be use in that certain process.

6.1 System Services Implementation Model

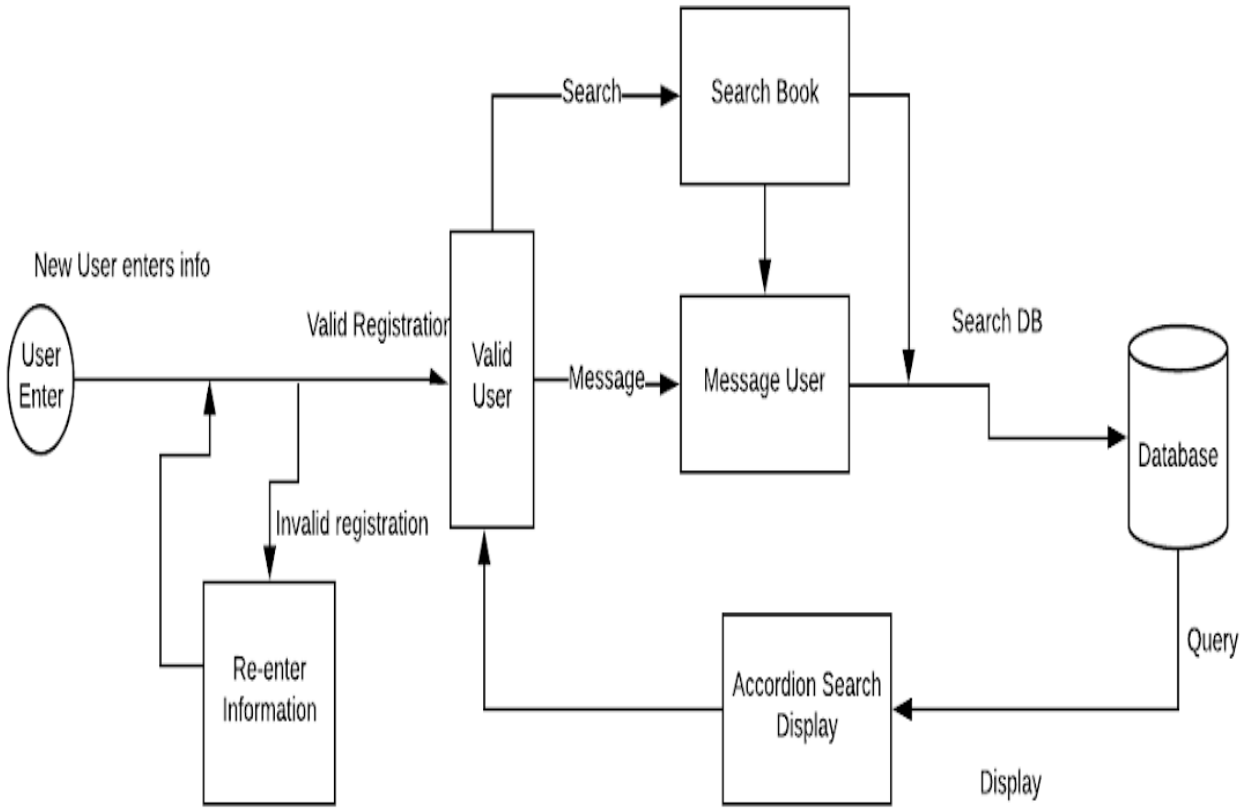
BUYER/SELLER Registration Case:
Use Case #1



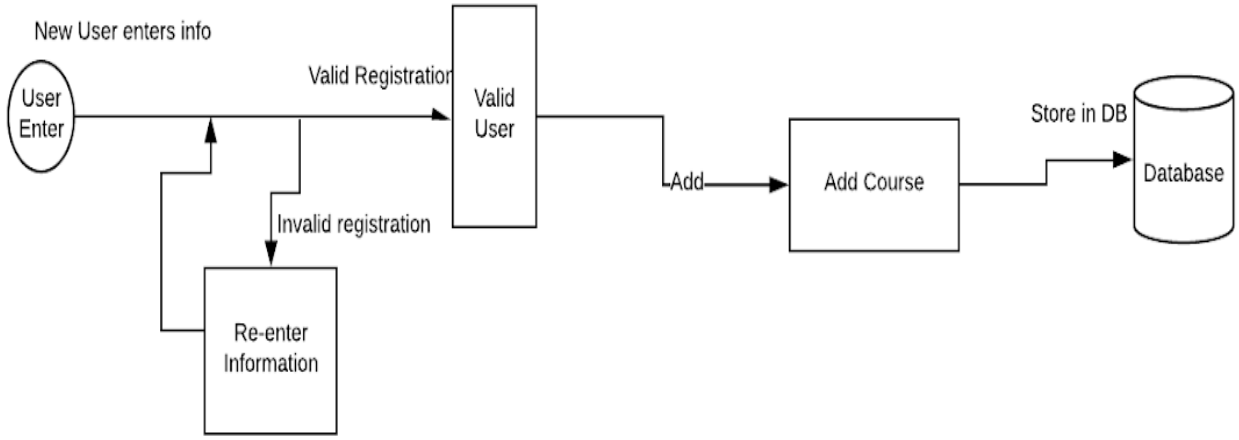
BUYER/SELLER/ADMINISTRATOR add/delete/change password BOOK case:
Use Case # 2,3,4



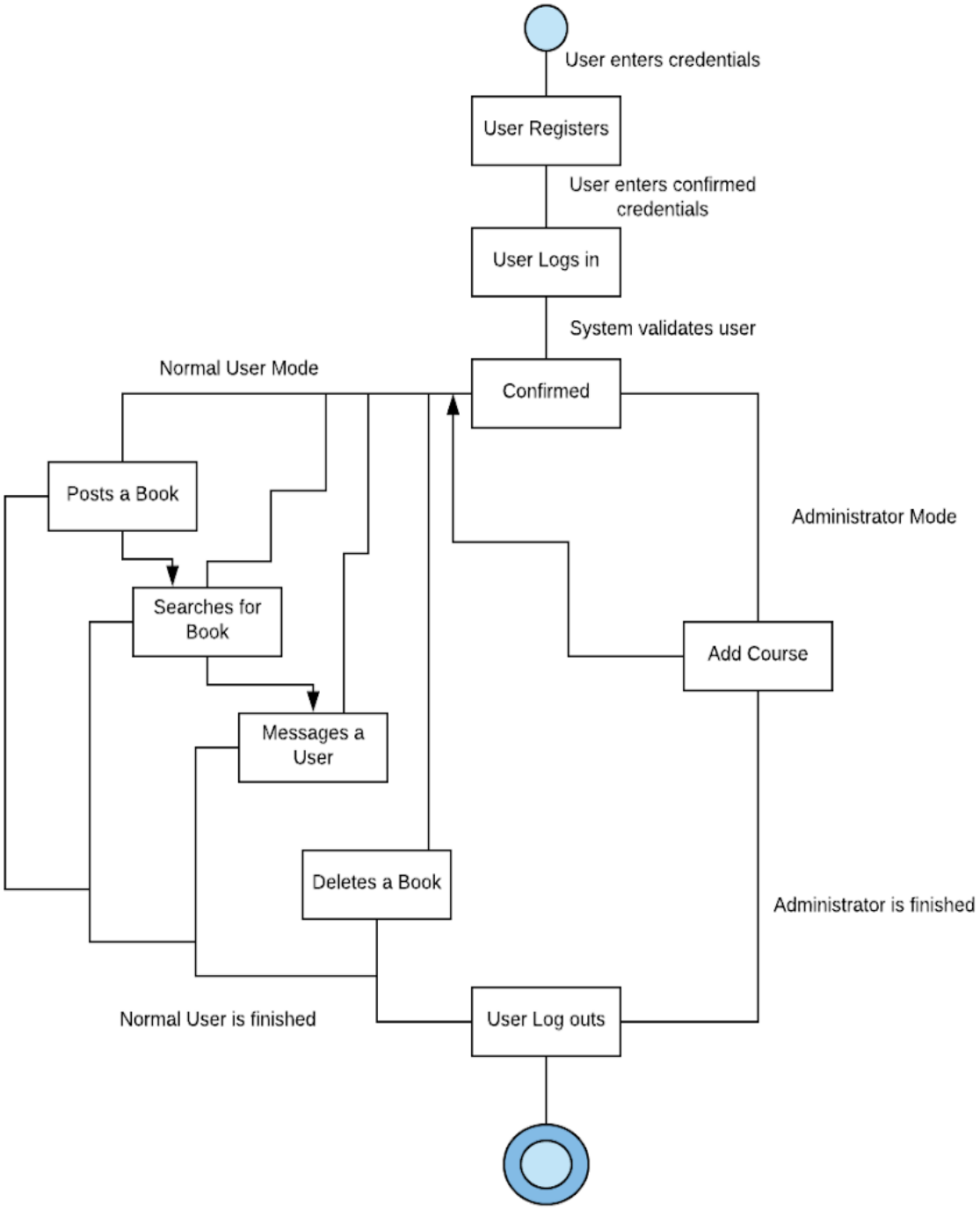
BUYER/SELLER/ADMINISTRATOR search/message BOOK results:
Use Case # 5,6



ADMINISTRATOR search/message BOOK results:
Use Case # 7



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.

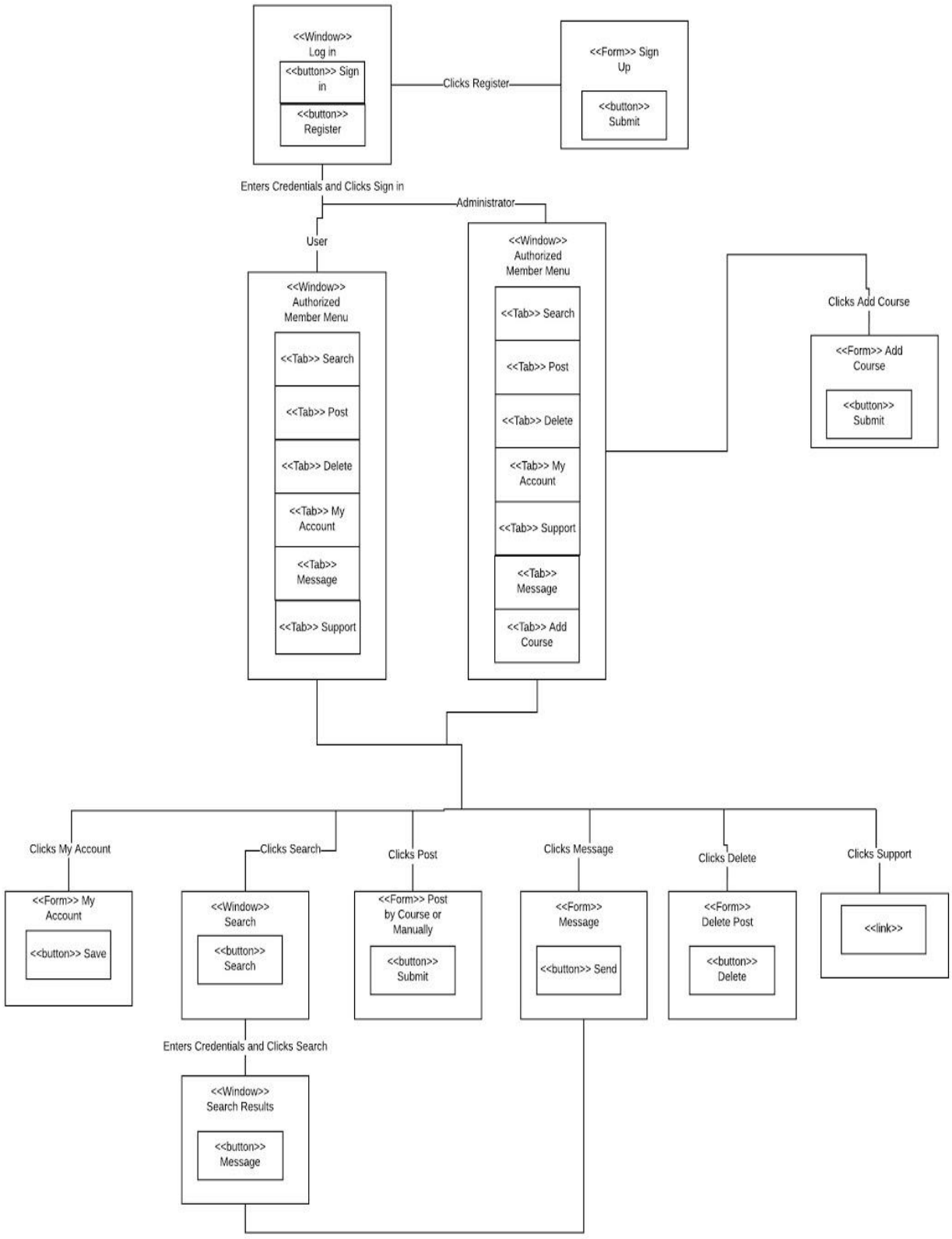
The following are the requirements for the user interface based on interviews and surveys conducted by our team:

- Availability
  - The system interface will be available online 24/7 except during routine maintenance.
- Organization
  - The system will have a simple side navigation for the main features of the system (My Account, Search, Post, Delete, Message, and Support).
  - In My Account Window, it will display the information gathered of the user when they first created the account. They will be able to change only their password; the username and e-mail address fields will not be able to be edited.
  - In the Search Window, it will allow users to enter textbook credentials. Once the user hits submit either in search by keyword, course, or all, 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 click a button to message the user.
  - In the Post Window, it will prompt users to a form that will allow users to quickly enter the textbook credentials either by course or entering the information manually.
  - In the Delete Window, it will prompt users with a list of all their posts. This section allows them delete posts for any reason.
  - In the Message Window, it will give the user the ability to enter a user they wish to send a message to and send them a message by clicking send.
  - In the Support Window, it will give the user the ability to get help using the application. In addition, it will give the user contact information of the team, if they have any further questions that have not been answered by visiting this window.
- 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 will validate the user email to check if it is an SPU email.
  - The system will not allow for users to enter the same username, as to not confuse other users. Usernames and emails must be unique and not stored in the database.

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

- The system will not be able to operate unless there are active users that have books for sale.
- The system will not allow anyone who does not have a valid SPU email to register for SPU Book Finder.
- The system will not have a chat feature, all messages will be made outside the applications, or done anonymously through the application.
- The system will only allow an administrator to add courses to the application.

7.2 Window Navigation Diagram



7.3 Screen Interface Design

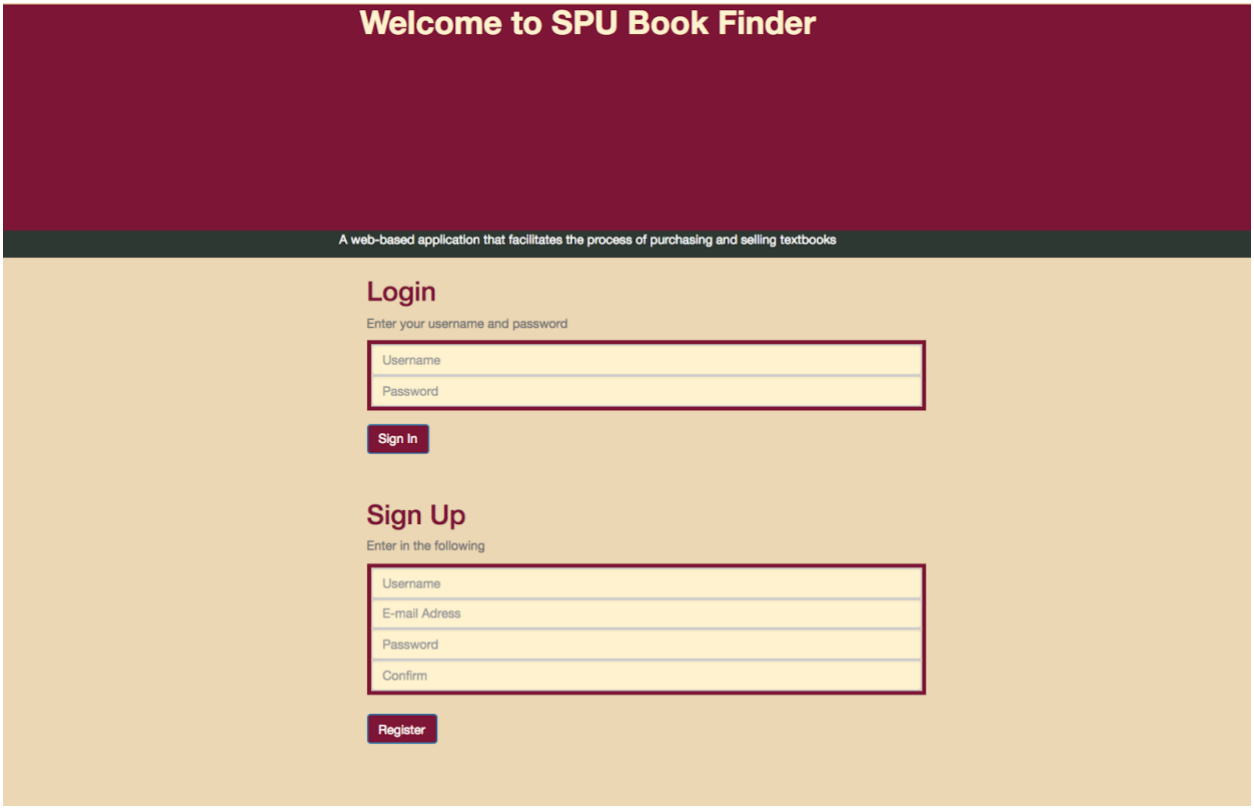


Figure 0. Log in screen. Option of logging in with credentials or creating an account.

Figure 1. To Create Account the user must fill out the information below. The username and email address must be unique. The email address shall only be a valid SPU email. A confirmation address with a link will be sent to the address provided. The user must confirm before being able to log in.

Figure 2. To log in to the system, the user will provide the fields above with a valid username and password.

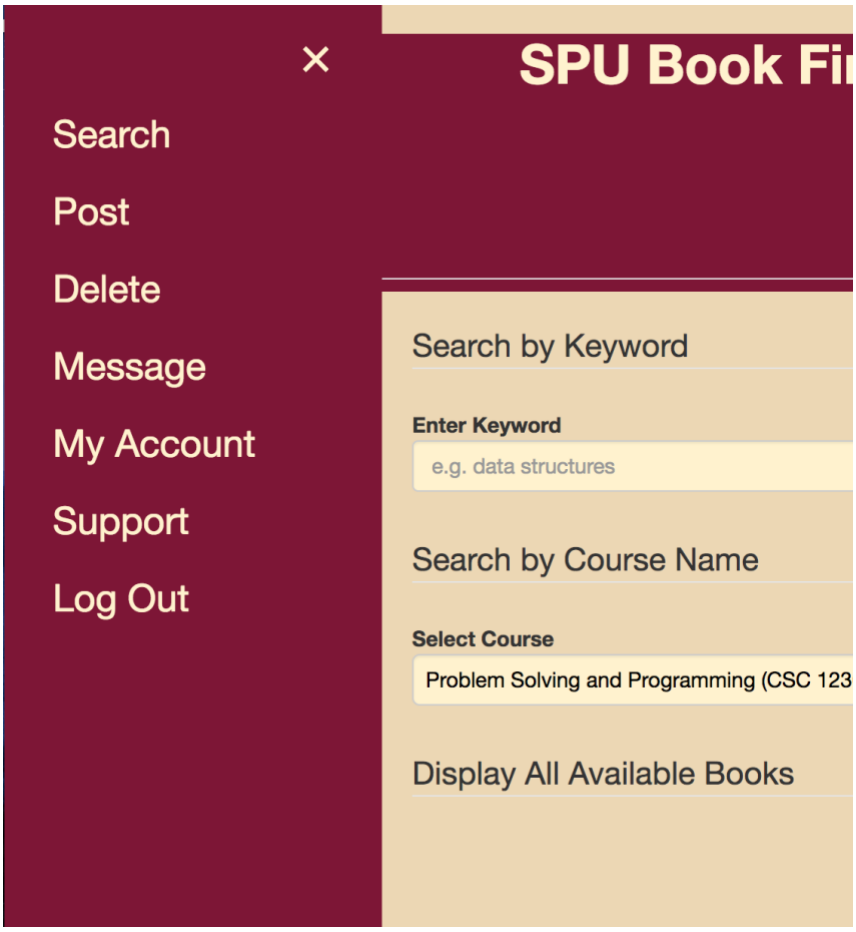


Figure 3. Once logged in, there will be a collapsible side window navigation. Depending on the user, administrative or normal user, the menu shall list navigation to the Search, Post, Delete, Message, My Account, Support, Add Course, and Log Out pages.

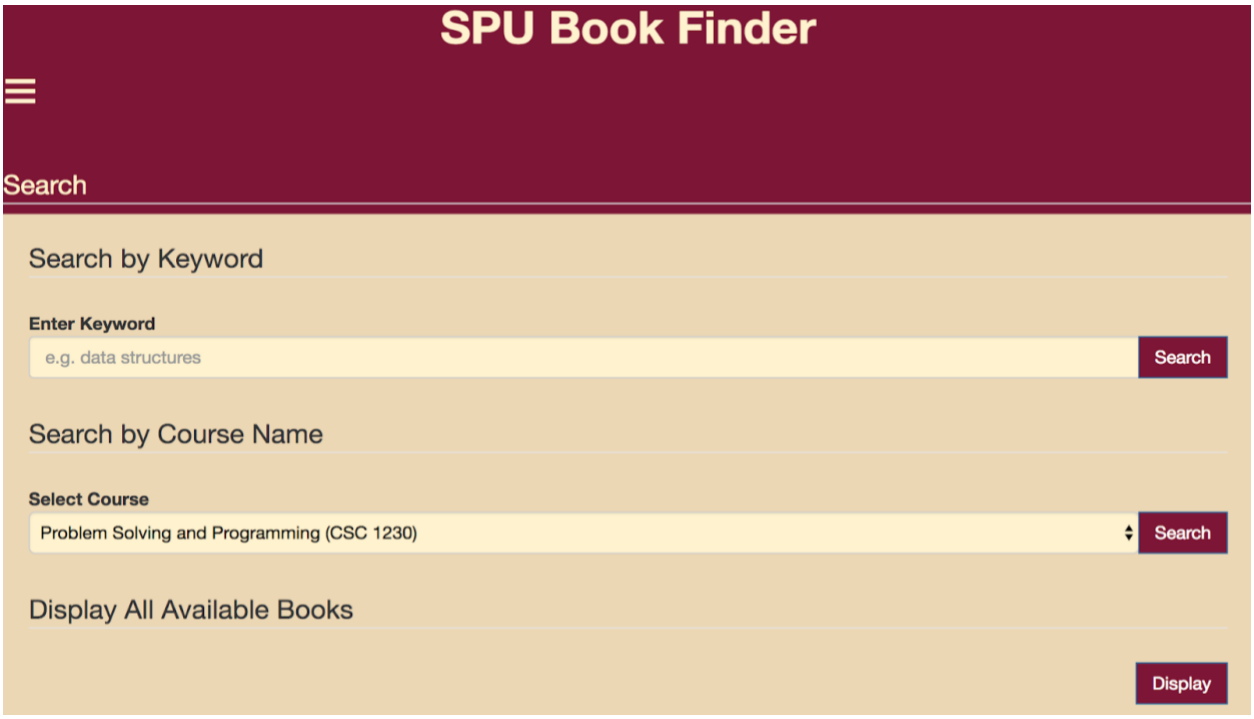


Figure 4. Once logged in, this is the landing page. This is also known as the Search Page. This page features three different types of searches: by keyword, course name, and displaying all.

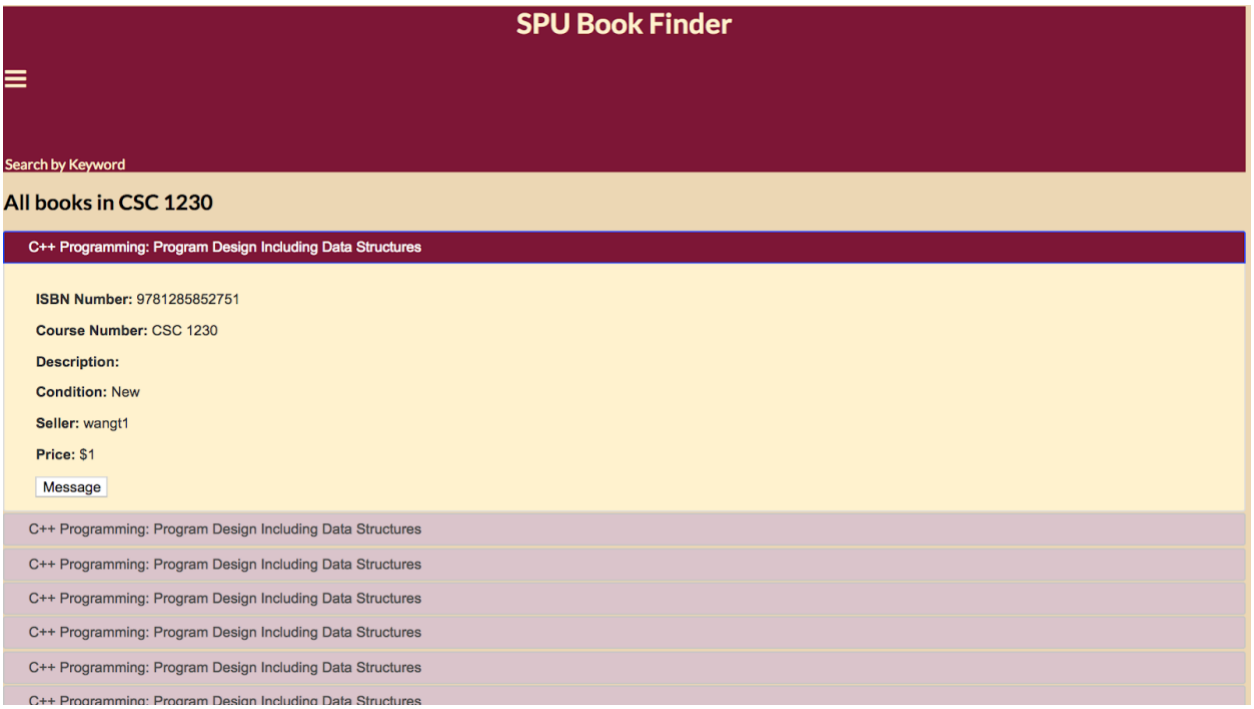


Figure 5. The window above shows the display of the search when any of the "Search" or "Display" buttons are pressed from Figure 4. Each book post, is seen in this accordion style. It displays: ISBN number, course number, description, condition, seller, price, and an option to message seller directly by pressing the "Message" button.

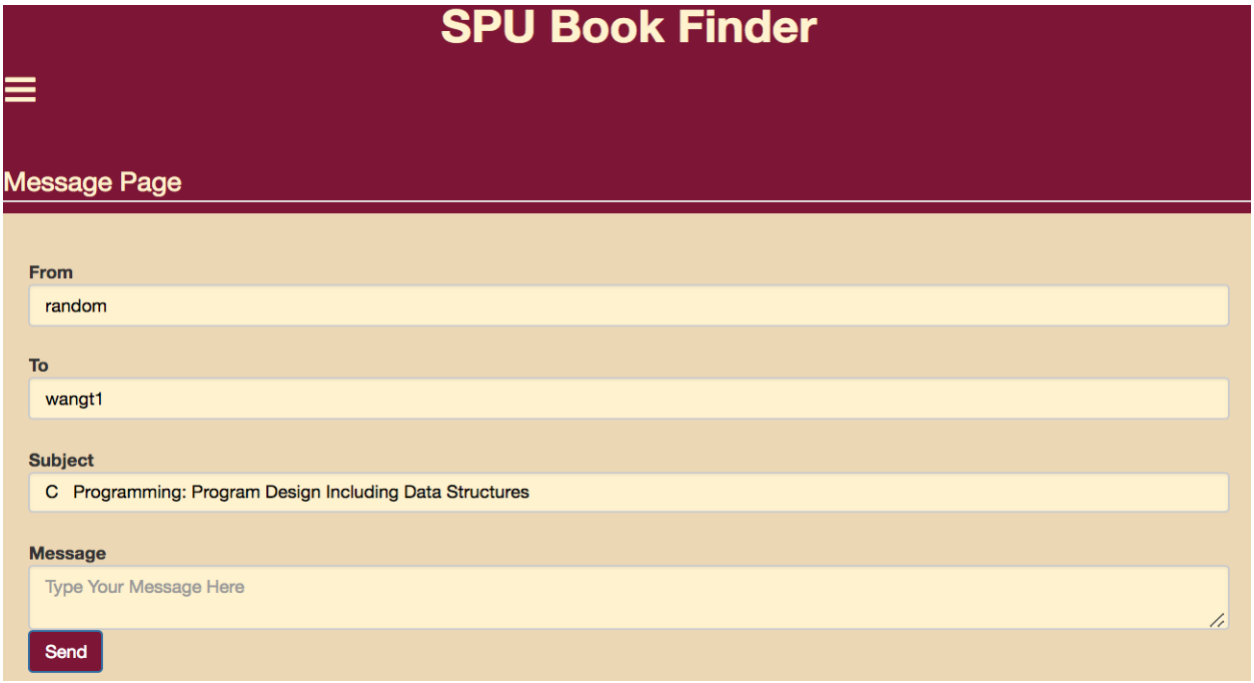


Figure 6. Window prompt when the "Message" button from Figure 5 or "Message" from the menu in Figure 3 is clicked. From field shall always be automated by the system. To, will be automated if pressed from Figure 5, but will be left blank when pressed from the menu. Subject and Message will be filled by the user. The user will click the "Send" button to send message.

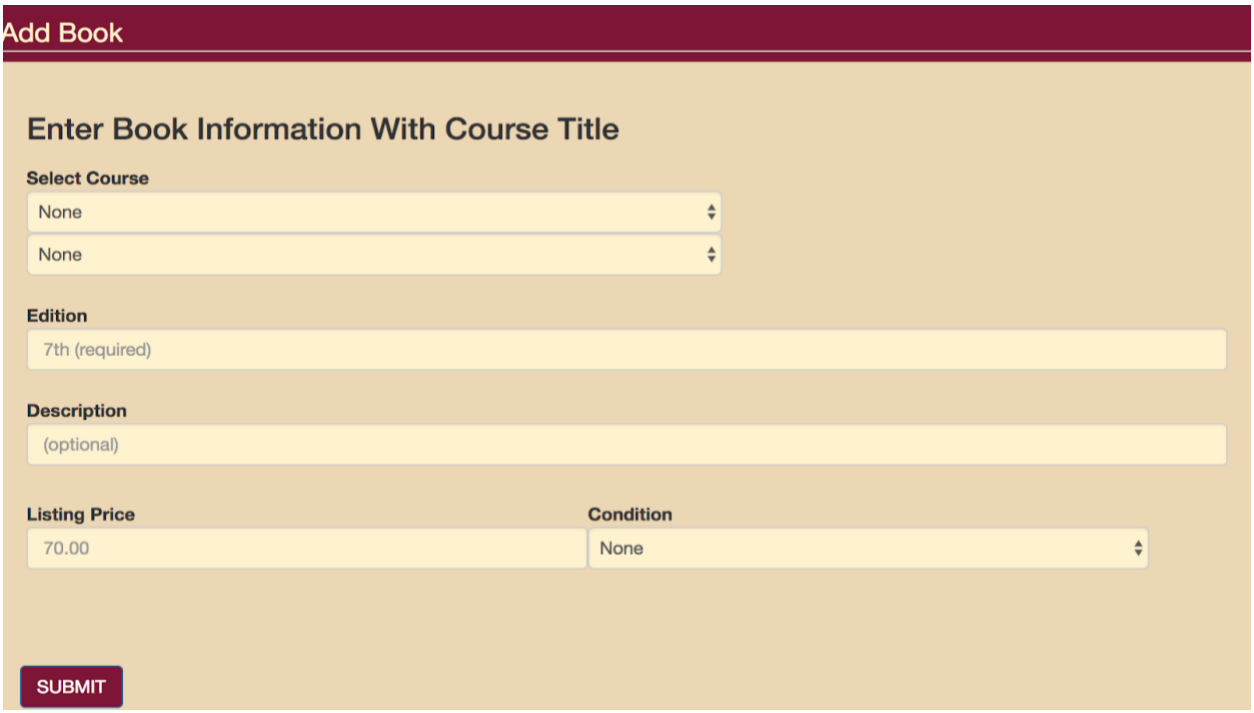


Figure 7. Window prompt when "Post" from the menu in Figure 3 is clicked. There are two ways to post a book, either by course or entering the information manually. Above shows fields to enter a book by course. The user clicks submit to enter the book into the database.

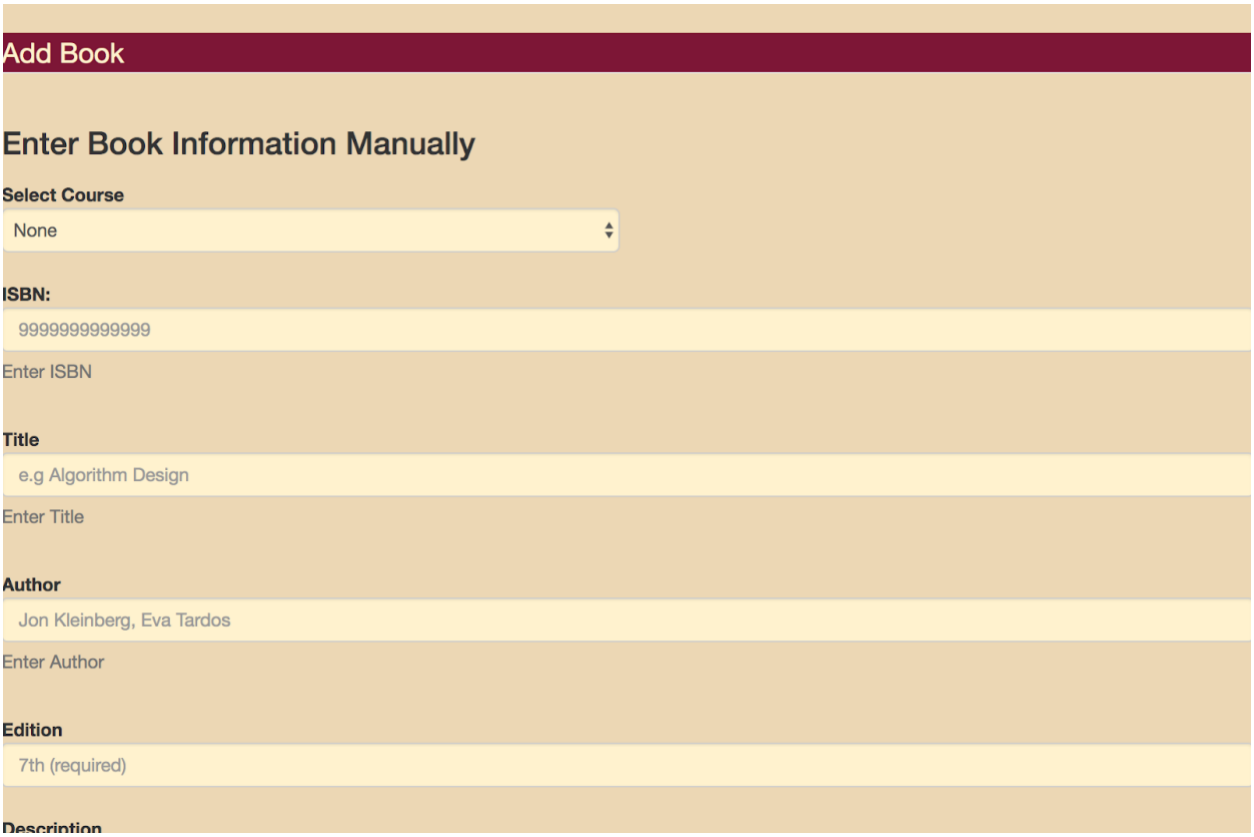


Figure 8. Window prompt when "Post" from the menu in Figure 3 is clicked. This is the second way to post a book. Above shows fields to enter a book manually. The user clicks submit below to enter the book into the database.

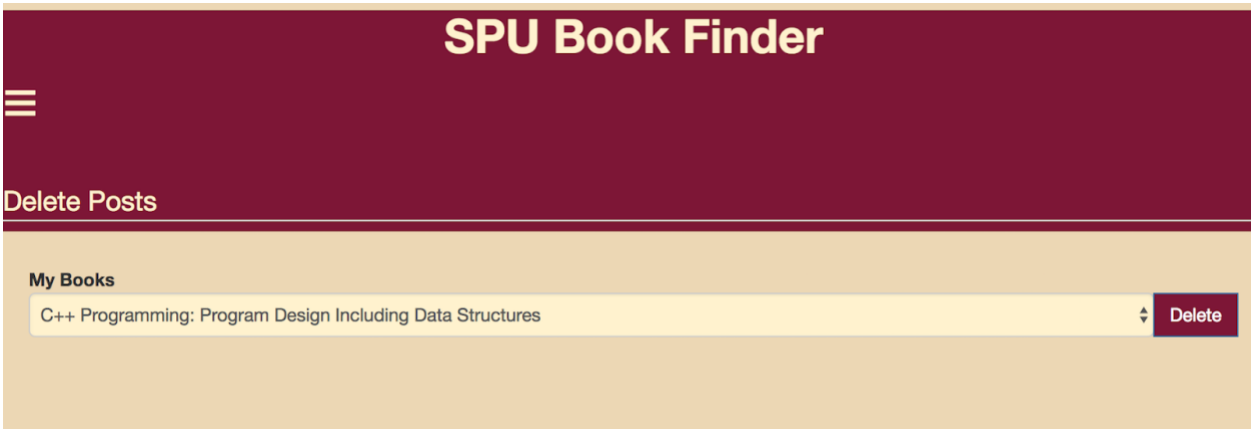


Figure 9. Window prompt when "Delete" from the menu in Figure 3 is clicked. The user can delete a book by opening the drop-down menu and selecting a book. Once the book is selected, the user can click the "Delete" button to delete the book post.

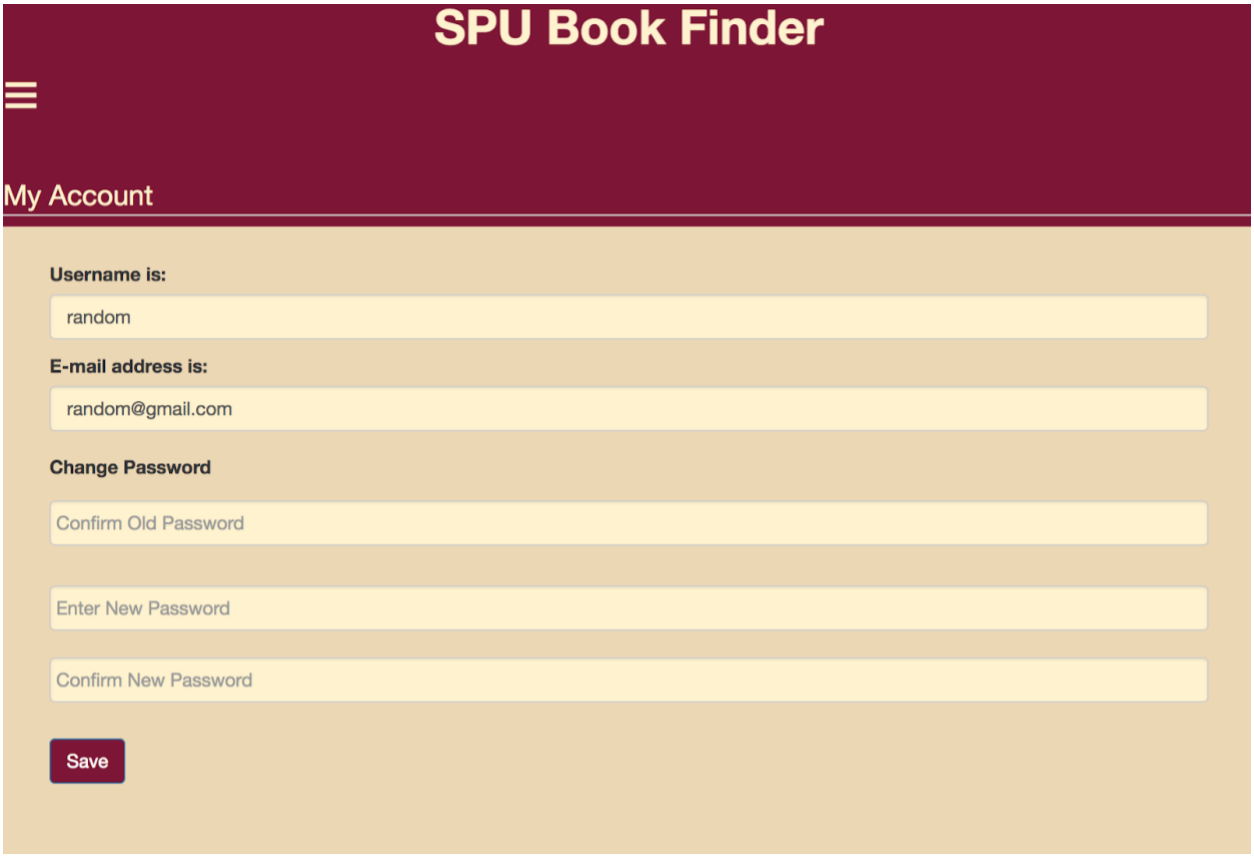


Figure 10. Window prompt when My Account clicked in Menu. This page will allow users to change their password. Username and email address fields cannot be altered by user.

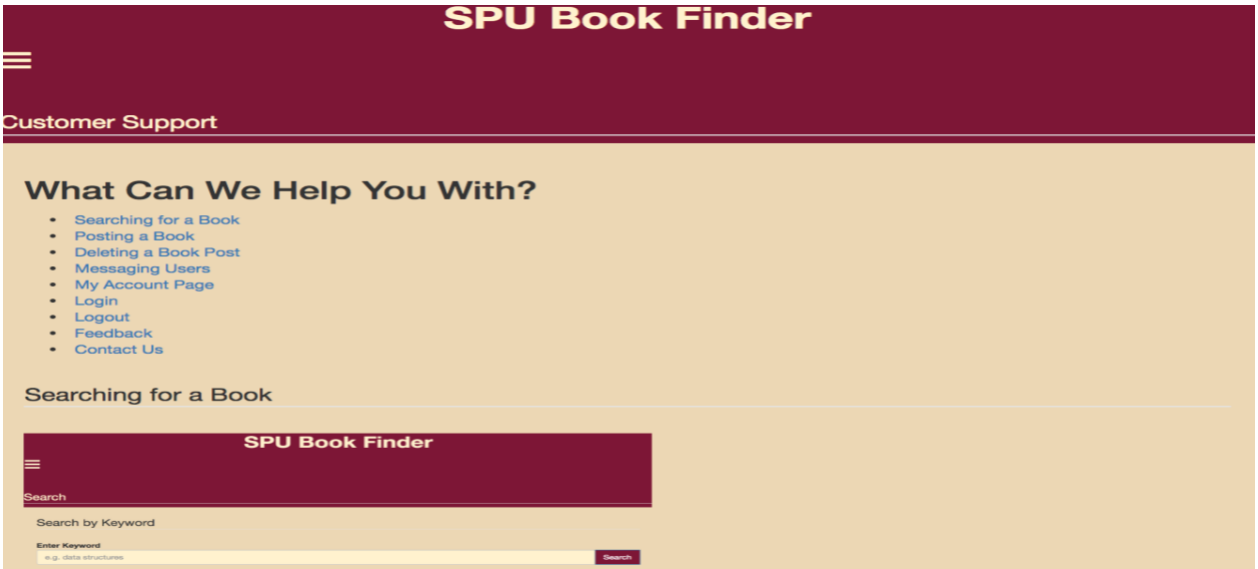


Figure 11. Window prompt when Support clicked in Menu. This page offers a manual on how to use the website. It allows the user to look for common features users have trouble with and see common steps that will help them out. If the page does not help, there is contact information of our team provided.

8.0 Development Plan

8.1 Development Schedule

Start Date	End Date	Task	Person(s) in Charge
3/31	4/6	Research on CRUD systems implementations	Cesar, Tianxing, Monica, Celena
3/31	4/6	Research and gather data to place into database	Celena, Monica
3/31	4/2	Create Database	Cesar
4/2	4/5	Populate Database	Celena
4/2	4/5	Create “flat” versions of all web pages • Log in • Create Account • My Account • Search • Post • Edit	Monica, Tianxing
4/11	4/13	Find place to hold database and website	Celena
4/6	4/13	PHP Log in/Account Feature	Monica, Tianxing
4/6	4/20	PHP Search Feature	Cesar, Tianxing
4/16	4/20	PHP Post Feature	Cesar, Celena
4/23	5/4	PHP Delete Feature	Cesar, Monica
4/23	5/4	Add CSS to make web pages look nicer	Monica, Celena
5/7	5/18	QA Testing Begins • Integration test: ○ All members are able to access system • Security:	Celena, Monica, Tianxing, Cesar

		○ Only SPU students with verified accounts can log in • Functionality Testing: ○ Ensure that all forms work such as posting books, editing posts, and searching for books ○ Ensure that data is written to the database • Compatibility Testing ○ Check that the system is compatible across different devices • Database Testing: ○ Make sure database integrity is maintained while creating, updating, or deleting data • Release Test ○ Anyone with a valid SPU email can access the system via the web	
4/30	5/30	User Testing and Survey	Celena, Monica
5/14	5/20	Another team conducting QA	Team Jakku testing Skateable
4/17	5/8	**Erickson Conference Poster** • Rough Draft Due 4/24 • Final Draft Due 5/1 • “Dry Run” in class Due 5/8	Celena, Monica, Tianxing, Cesar
5/21	5/31	QA Testing Continue and Finish	Celena, Monica, Tianxing, Cesar
	5/31	Deliver finished product	Celena, Monica, Tianxing, Cesar