# Assignment 4

**Ethan Staiman**

***Pathname:*** /locale/home/sysadmin/bazaar

**GitHub Repository:** https://github.com/EStaiman/TCNJ-Bazaar

***Use Case Descriptions:***

Use Case: User posts textbook

Primary Actor: Student/User

Goal in Context: To post a textbook for sale

Preconditions: User is logged in

Trigger: User wants to post textbook

Scenario:

1. User clicks the Sell Textbook Button
2. They enter the textbook name, edition, condition, class(es) used for, and asking price
3. They click post
4. They then are redirected to the listing

Exceptions:

1. Information in one or more fields is incorrect or blank – The user stays on the page; the website indicates what fields were incorrect and what should be entered
2. They select cancel instead of post – They are taken back to home

Priority: High

Frequency of Use: Frequent

Open Issues:

1. What would be invalid info for specific fields?

Use Case: User searches for textbook

Primary Actor: Student/User

Goal in Context: To browse for a textbook

Preconditions: User is logged in

Trigger: User wants to search for a textbook

Scenario:

1. User clicks the Search Textbook Button
2. They select what criterion they want to search by (textbook name, edition, condition, class(es) used for, asking price, etc.)
3. They enter in what they want to find
4. They click search
5. They then view the listings based off of their search criteria

Exceptions:

1. Information in one or more fields is incorrect or blank – The user stays on the page; the website indicates what fields were incorrect and what should be entered
2. They are no textbooks matching the searched terms – The user stays on the same page with a message indicating that no textbooks matched their search results

Priority: High

Frequency of Use: Frequent

Open Issues:

1. What would be invalid info for specific fields?

Use Case: User places offer for textbook

Primary Actor: Student/User

Goal in Context: To place an offer for the textbook

Preconditions: User is logged in

Trigger: User wants to place an offer for a textbook

Scenario:

1. User clicks the Search Textbook Button
2. They select what criterion they want to search by (textbook name, edition, condition, class(es) used for, asking price, etc.)
3. They enter in what they want to find
4. They click search
5. They then view the listings based off of their search criteria
6. They click on a textbook that they want to buy and are redirected to that textbook page
7. They click send offer

8. They enter in the offer and click send
9. They are brought back to the home screen with a message appearing saying that their offer was sent

Exceptions:

1. Information in one or more fields is incorrect or blank – The user stays on the page; the website indicates what fields were incorrect and what should be entered
2. They are no textbooks matching the searched terms – The user stays on the same page with a message indicating that no textbooks matched their search results
3. The offer is nonnumeric – The user stays on the same page, and the website indicates that they need to enter in numbers and '.' only (No $)

Priority: High

Frequency of Use: Frequent

Open Issues:

1. What would be invalid info for specific fields?
2. Whether we should allow messages containing more then the offer, like asking questions about the textbook, or have that all be done through email outside of the website.

Use Case: User edits textbook listing

Primary Actor: Student/User

Goal in Context: To edit a textbook listing

Preconditions: User is logged in

Trigger: User wants to edit their listing

Scenario:

1. User goes to their profile
2. They click on the listing they want to edit
3. They selected edit
4. They change any of the fields
5. They click save
6. They then are redirected to the listing that has been changed

Exceptions:

1. Information in one or more fields is incorrect or blank – The user stays on the page; the website indicates what fields were incorrect and what should be entered. Changes won't be saved unless all information is valid.
2. They select cancel instead of post – They are taken back to the original listing under their profile

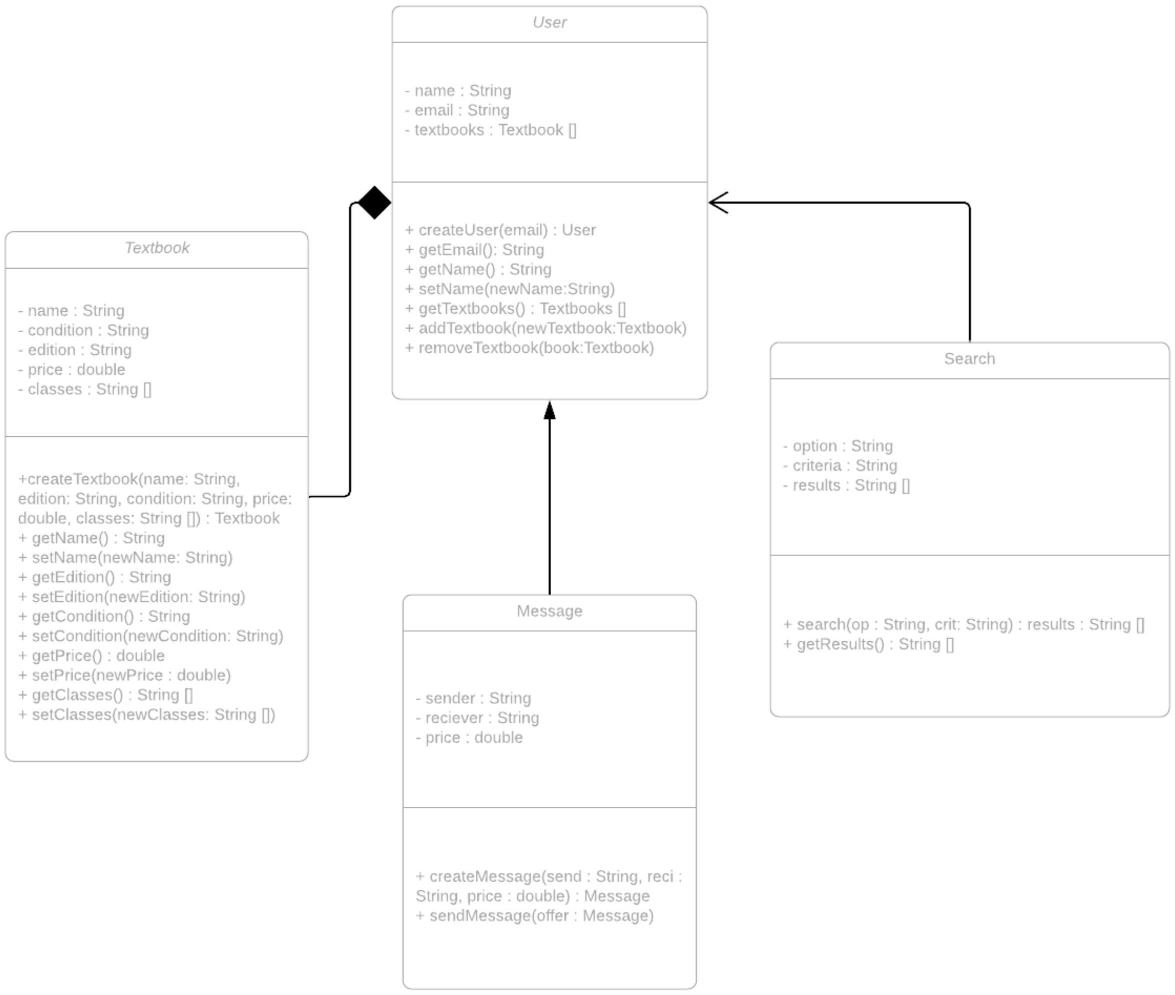
Priority: Medium-High

Frequency of Use: Semi-Frequently

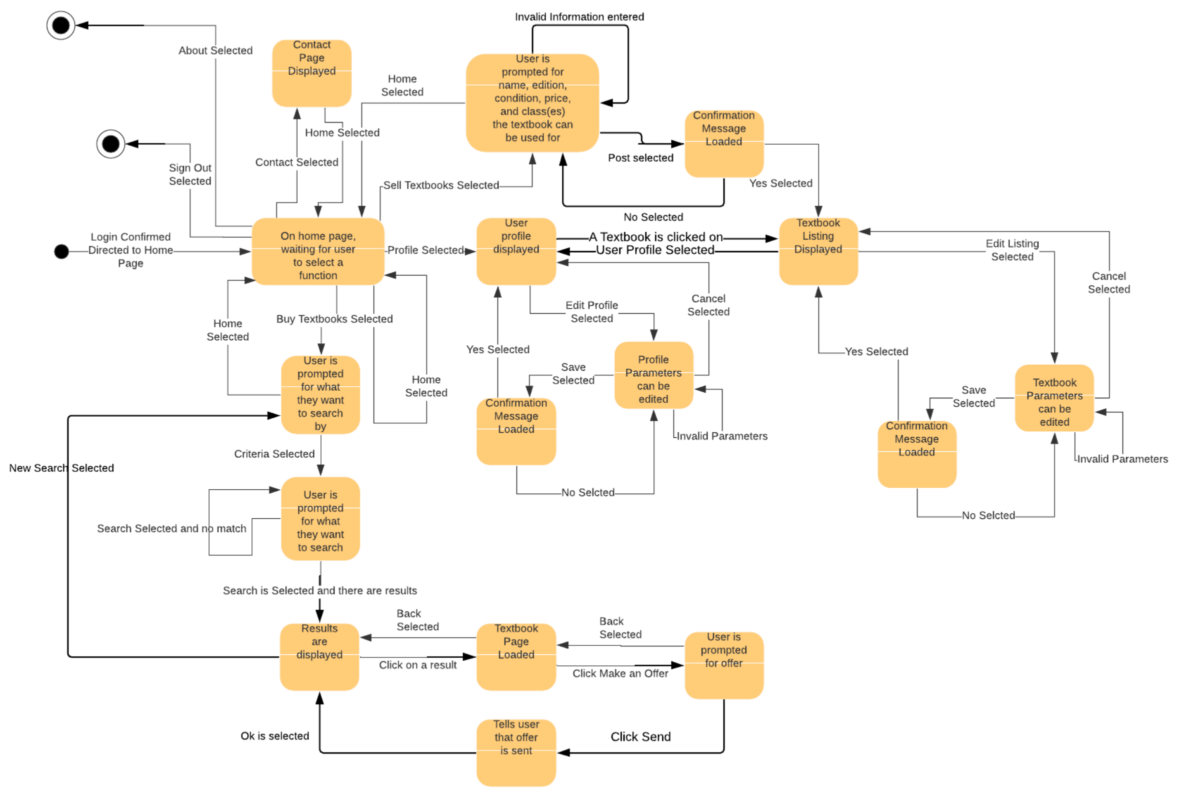
Open Issues:

1. What would be invalid info for specific fields?

***Detailed Design Class Diagrams:***

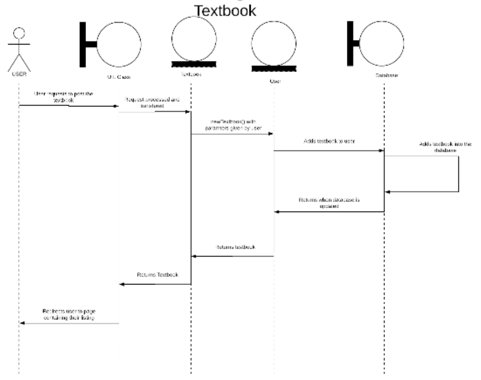

User
- name : String
- email : String
- textbooks : Textbook []
+ createUser(email) : User
+ getEmail(): String
+ getName() : String
+ setName(newName:String)
+ getTextbooks() : Textbooks []
+ addTextbook(newTextbook:Textbook)
+ removeTextbook(book:Textbook)
Textbook
- name : String
- condition : String
- edition : String
- price : double
- classes : String []
+createTextbook(name: String, edition: String, condition: String, price: double, classes: String []) : Textbook
+ getName() : String
+ setName(newName: String)
+ getEdition() : String
+ setEdition(newEdition: String)
+ getCondition() : String
+ setCondition(newCondition: String)
+ getPrice() : double
+ setPrice(newPrice : double)
+ getClasses() : String []
+ setClasses(newClasses: String [])
Search
- option : String
- criteria : String
- results : String []
+ search(op : String, crit: String) : results : String []
+ getResults() : String []
Message
- sender : String
- reciever : String
- price : double
+ createMessage(send : String, reci : String, price : double) : Message
+ sendMessage(offer : Message)

## *Statechart:*

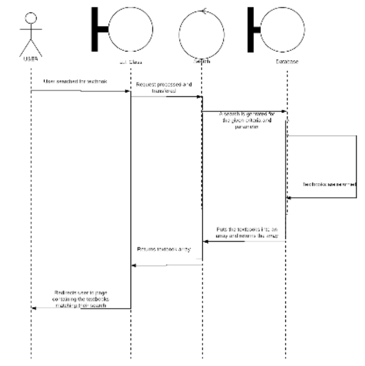


## *Detailed System Sequence Diagrams:*

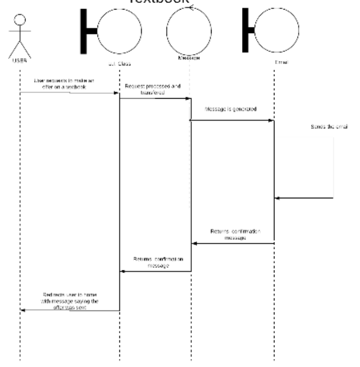
## Posting a Textbook


USER
U.I. Class
Textbook
User
Database
User requests to post the textbook
Request processed and transfered
newTextbook() with parameters given by user
Adds textbook to user
Adds textbook into the database
Returns when database is updated
Returns textbook
Returns Textbook
Redirects user to page containing their listing

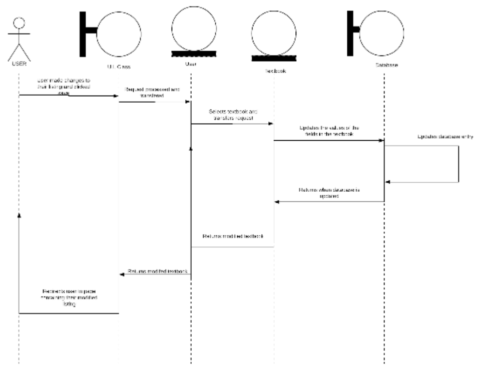
## Searching for a Textbook


USER
U.I. Class
Search
Database
User searched for textbook
Request processed and transfered
A search is generated for the given criteria and parameter
Textbooks are returned
Puts the textbooks into an array and returns the array
Returns textbook array
Redirects user to page containing the textbooks matching their search

## Make an Offer on a Textbook

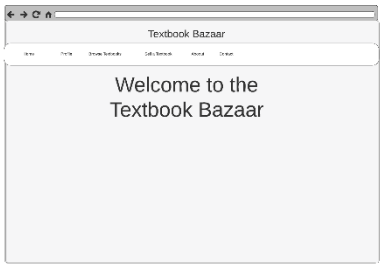

USER
U.I. Class
Message
Email
User requests to make an offer on a textbook
Request processed and transfered
Message is generated
Sends the email
Returns confirmation message
Returns confirmation message
Redirects user to home with message saying the offer was sent

## Edit a Listing

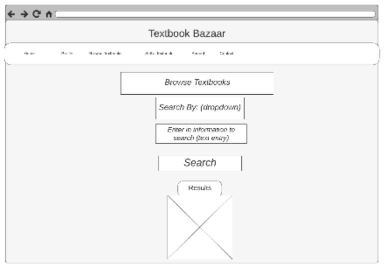

USER
U.I. Class
User
Textbook
Database
User made changes to their listing and clicked save
Request processed and transfered
Selects textbook and transfers request
Updates the values of the fields in the textbook
Updates database entry
Returns when database is updated
Returns modified textbook
Returns modified textbook
Redirects user to page containing their modified listing

***UI:***

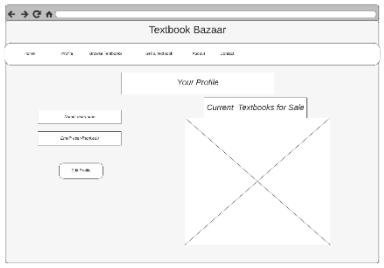
Welcome
Screen



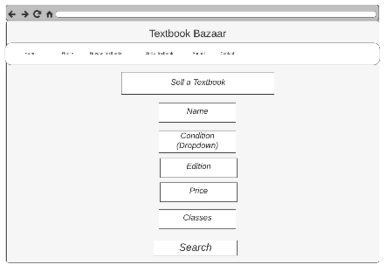
Browsing
Textbooks
Screen



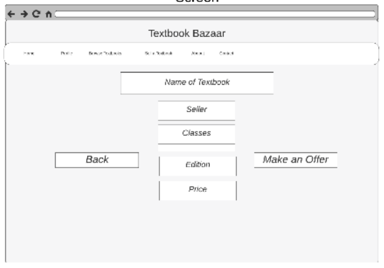
User
Profile
Screen



Selling a
Textbook
Screen



Viewing a
Textbook (you
don't own)
Screen



My UI will have visible and intuitively labeled buttons and text boxes/dropdowns to maximize user experience. These buttons will give immediate feedback, which follows the principle of visibility. Additionally, the intuitive design, placement, and headers for the UI elements follows the principle of affordance. My design is constant, as there is a consistent header on every page and there will be a consistent color/graphical design. Frequent users will be able to utilize the navigation bar so they can shortcut throughout the site. The buttons will be responsive to touch and give feedback. There will be a clear indication when the page is done loading/performing an action, which will result in the page clearly changing. There will be simple error handling in whenever the user can input data, there will be a check to make sure the data is valid for that field. If not, the user will be brought back to the page and there will be a message indicating what field was invalid and what should be entered in said field. Anytime a user is editing something, they will always have the option of not saving their changes and undoing what they have done. The easy navigability of the site allows the user to feel in control of the site. Finally, since all the buttons are so intuitive and easily accessible from any page with the navigation bar, users will not be forced to memorize where things are.

***Design Criteria:***

*Modularity and Encapsulation:* All of my data and methods are contained within my classes. These classes allow the data to be available only to the class itself. This implements information hiding. Additionally, the methods I defined can be used in multiple cases. This allows the code to be reused.

*Elegance and Efficiency:* The algorithms I will be using will query the database for information. This will simply just search every textbook in the database, which will be O(n) time. Since we are searching with non-numerical criteria, a binary search is not possible. A simple O(n) search is very easy to implement, and its elegance is in its simplicity.

*Data Structure:* I will be using Textbook objects to store the textbook information. There are arrays of Textbook objects that store the textbooks for a specific student and as the end result of a search. The Textbook object is the most appropriate way of storing the textbook information as it allows for information hiding. Storing the Textbook objects in arrays within a user object is the most appropriate because it allows for quick access to the Textbook objects without the larger overhead of linked lists/trees.

***Test Case Design:***

For unit testing, I will be verifying that each individual function works. I will test to see if I can browse for a textbook, view a textbook, make an offer, put a textbook up for sale, edit the listing, and all other functions. Each function will be tested independently, and I will make sure each unit functions as intended. For integration testing, I will start testing the units together.

For example, if user A posts a textbook, can user B see that textbook? If the units were functioning separately but not together, then I will have to find how one unit is affecting the other and vice-versa. Finally, for system testing I will make sure the system as a whole is functioning. Instead of testing a couple of units being integrated, I am testing that the entire system can be integrated. The end result of this round of testing is that the website feels like a functioning website that can be began to be used.

I will be using RSpec Rails as my testing tool. Since I had done my assignment 2 on testing, and I worked a lot with RSpec, I looked into seeing if it would work with rails. I ended up finding RSpec Rails, which that extends RSpec into rails with specs for models, specs, and other things in the rails framework.

| **Functionality Tested** | **Inputs** | **Expected Outputs** | **Actual Outputs** |
|---|---|---|---|
| Any place where a user can enter in data, it will catch invalid data being entered, such as alphabetic characters for the price or leaving a required field blank. Both for creating a textbook and editing a textbook/profile. | For price, test both letters and symbols, only thing that should be accepted is 0-9 and '.'. Also test entering in nothing in one field at a time, multiple fields at a time, and all fields. | Error messages explaining where the error(s) occurred and what should be entered in those fields. Leaving the page without entering in valid data will discard changes when editing and discard the listing if creating a textbook. | |
| You can search for a textbook by any criteria | For each criterion, enter in a valid keyword that will generate a match | The textbook is displayed | |
| When searching for a textbook, tell the user if no results are found | Enter in a keyword that you know will not generate a match with the selected criteria. | A no results error message is displayed, and you can create a new search | |
| You can create a textbook to sell | You enter in valid criteria for each field | You are brought to the page for your listing | |
| You can edit your listing | You selected edit on the listing and changed some values, making sure they are still valid | The listing is changed | |
| You can make an offer | You searched for and found a textbook, | The user who posted the listing receives a | |

| | | | |
|---|---|---|---|
| | clicked the textbook, clicked make an offer, entered in a valid number, and selected send. | message from the original user with the correct offer. | |

***Open Source Maintenance and Communication:***

The best place to communicate bugs and organize issues is through the GitHub page. There users can create issues for anything they feel needs to be implemented and alert others of any bugs encountered and start working towards a solution for them. Similarly, GitHub is ideal for discussing any future plans. Users can create a local copy of a branch and begin to test/develop. All code should be done with good programming practices. No global variable usage, error handling any time the user enters data in, short yet informative variable names, etc. Any new pages should graphically be in line with the current pages. All changes will be verified by me before they can be accepted. If it's a small change/bugfix, a simple message on what was fixed should be sufficient. If new functionality is added, I would have to more rigorously go through and make sure everything is working as intended. If significant graphical changes occur, I would create a poll of some sort on the GitHub page and ask people what they think. If people are supportive of the new UI, it will be implemented. If the change is for efficiency, such as changing an algorithm to have a lower time complexity, it would have to be verified by 100s of repetitions of the old and new to see which is faster. I think because this is a small project having me vetting additions is feasible and helps keep the project in line with my original ideas.