University of New Hampshire Textbook Exchange

DEPARTMENT OF COMPUTER SCIENCE

CONNECTING STUDENTS ACROSS CAMPUS FOR THE PURPOSE OF BUYING AND SELLING TEXTBOOKS

Stephen Chambers

Scott Cypher

Jason Gilneas

Nicholas Sjostrom

Abstract: The goal of this project is to develop a textbook exchange application that enables students to sell textbooks to other students. The application will be developed in *Java* using the *Eclipse Integrated Development Environment* and will utilize different software tools such as the *Amazon Product Advertising API*, the open source database *MySQL*, the *Apache Commons Development Libraries* and *Swing*. An online server will be used to store all user and book information allowing students to easily search for a textbook and contact each other.

Iteration 1 Effort-Breakdown Table

	Team Member Name			
	Stephen Chambers	Scott Cypher	Jason Gilneas	Nicholas Sjostrom
Project Management (5%)	25%	25%	25%	25%
Report: Requirements Specification (15%)	25%	25%	25%	25%
Report: Object Model (20%)	25%	25%	25%	25%
Report User Interface Model (10%)	25%	25%	25%	25%
Report Preparation (5%)	25%	25%	25%	25%
Implementation: Coding (15%)	25%	25%	25%	25%
Implementation (20%)	25%	25%	25%	25%

Iteration 2 Effort-Breakdown Table

	Team Member Name			
	Stephen Chambers	Scott Cypher	Jason Gilneas	Nicholas Sjostrom
Project Management (5%)	25%	25%	25%	25%
Report: Requirements Specification (15%)	25%	25%	25%	25%
Report: Object Model (20%)	25%	25%	25%	25%
Report User Interface Model (10%)	25%	25%	25%	25%
Report Preparation (5%)	25%	25%	25%	25%
Implementation: Coding (15%)	25%	25%	25%	25%
Implementation (20%)	25%	25%	25%	25%

Iteration 3 Effort-Breakdown Table

	Team Member Name			
	Stephen Chambers	Scott Cypher	Jason Gilneas	Nicholas Sjostrom
Project Management (5%)	25%	25%	25%	25%
Report: Requirements Specification (15%)	25%	25%	25%	25%
Report: Object Model (20%)	25%	25%	25%	25%
Report User Interface Model (10%)	25%	25%	25%	25%
Report Preparation (5%)	25%	25%	25%	25%
Implementation: Coding (15%)	25%	25%	25%	25%
Implementation (20%)	25%	25%	25%	25%

Table of Contents

1. Introduction

1.1. Problem Definition

At the beginning of each semester, students are required to buy textbooks for their classes. Additionally, returning students have leftover textbooks from the previous semester that they no longer have use for.

Currently, the only local options students have to purchase or sell textbooks are the *Durham Book Exchange* and the *University of New Hampshire Book Store*. At both of these locations the books being sold are priced at high rates while the books the store is buying are priced well below what the student originally paid for them. Because of this, students are left spending a lot of money to purchase textbooks while only getting a minimal amount of money back for their old textbooks.

The *University of New Hampshire Book Exchange* is an application designed to connect students across campus for the purpose of buying and selling textbooks. By removing the middle man, this application will allow students to save money while purchasing textbooks and get more money back when selling textbooks.

1.2. Project Overview

The University of New Hampshire Book Exchange is a portable application capable of supporting multiple concurrent users. The application has been developed using the *Eclipse Integrated Development Environment* and is based primarily in *Java 1.7*. The graphical user interface for the application was developed using the Swing framework. The application utilizes the open source database *MySQL* in order to store relational database tables holding information for book, user, transaction and customization information. When accessing these tables, the application queries an online server to update a certain table depending on the user's operation in the interface. Because an online server is used, all tables are able to be accessed and modified live from any computer with a valid internet

connection. In order to retrieve book information such as an image of the book or the book's authors, the *University of New Hampshire Book Exchange* uses the *Amazon Product Advertising API*. Additionally, the *Apache Commons Library* is used to send e-mails to different users. Scenarios where e-mails are sent include the registration process, when buying a book, and when modifying your account information.

1.3. Release Notes

Version 1.0

The user is able to register to the database, sell a book, and search for an already posted book.

Known Issues:

1. Users are able to search the database without being logged in.
2. The same Wildcats ID can be used by multiple accounts.
3. If a user is deleted, their books remain in the database.
4. Book information does not need to be valid to be entered into the database.

Version 1.1

In addition to previous functionality, the user is able to customize their book tables, display a how to guide to help them use the application, and query Amazon in order to display a book's cover image, title, author, and other relevant information to the sale. Administrative privileges have also been added to this version. An administrator has the ability to modify a user's account information, delete a user, modify a book, and delete a book from the database.

Known Issues:

1. Invalid ISBNs cause unexpected application behavior.
2. The email regular expression during the registration process and the book price regular expression during the sell book process do not behave correctly.

3. Querying a large number of Amazon requests in a short time period causes unexpected behavior.
4. JTableHeader color is not updated after customization occurs.
5. Administrative modification does not behave correctly under all circumstances.

Version 2.0

This revision consisted of a complete refactoring of *Version 1.1*. Specifics about the refactoring process can be found in Section 3.5, Refactoring. In addition to previous functionality, the user is able to buy books, view their transaction history, modify their account information, and retrieve a lost password. As part of the refactoring process, *Version 2.0* has also been revamped from an interaction and aesthetic perspective. The user is provided with feedback for every action, even if the action is unsuccessful in order to reassure the user their action is being taken into consideration by the system or that their request has been successfully processed. The navigation system for the application has also been updated to be more graphic oriented than text oriented. In terms of aesthetics, the interface is more visually pleasing due to the improved style, font, and color schemes.

Known Issues:

1. An unstable internet connection can cause unexpected behavior.
2. Table columns with numerical values do not sort properly.
3. Passwords are handled without encryption by the application

2. Software Requirements

2.1. Use Cases

The following section is a general abstraction of the current application. This section focuses on following use cases in the following order:

1. Register User: The user is able to successfully store their account information in the database.
2. Sell Book: The user is successfully able to post a book to the database.
3. Buy Book: The user is successfully able to search for and initiate a transaction for a book that is currently in the database.
4. Update Account The user is able to update their account information (such as name and e-mail) currently in the database.
5. Manage User The administrator is successfully able to modify and delete user account information stored in the database.
6. Edit Book User is able to successfully edit information regarding a book they have already posted.
7. Customize Page The user is able to successfully save custom font and color information that change how the application's tables display.
8. View Transactions The user is able to view the books that they requested to buy and requests for books that they are selling.

2.1.1. Use Case Diagram

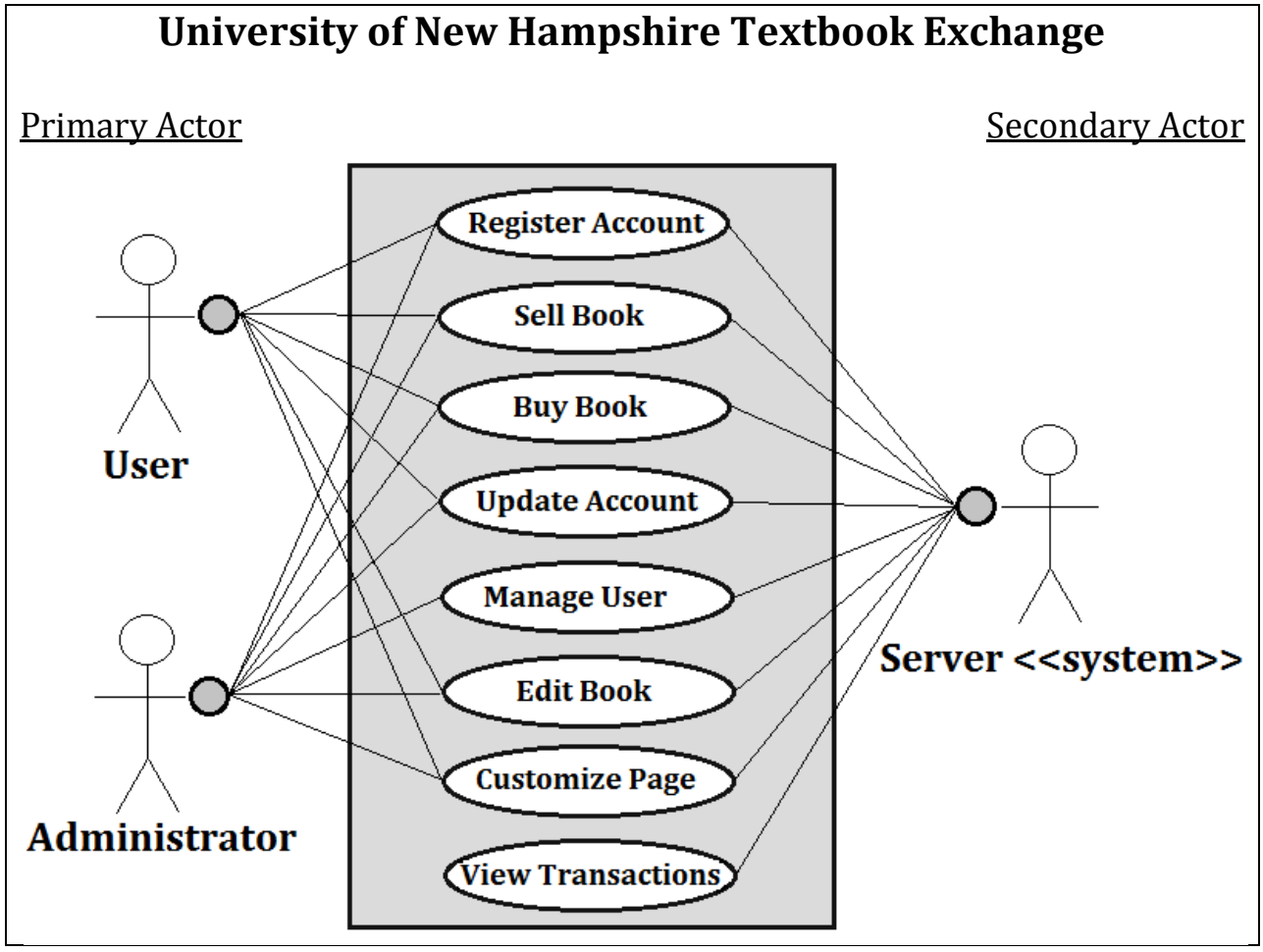


Figure 2.1: University of New Hampshire Textbook Exchange Use Case

Use case diagram for Register Account, Sell Book, Buy Book, Update Account, Manage User, Edit Book, Customize Page, and View Transactions use cases. The primary actors are the user and administrator and the secondary actor is the server. Because all information is stored in a server, the user and administrator must communicate with the server in order to access and update user, transaction, customize, and book information.

2.1.2. Fully Dressed Use Cases

1. Register User	
Use Case Section	Description
Use Case Name	Register User
Scope	*University of New Hampshire Textbook Exchange*
Level	User-Goal
Primary Actor	User
Stakeholders and Interests	User: Wants to register for the system in order to sell and buy textbooks. Server: Need to store information for the client.
Preconditions	User has a legitimate UNH email address.
Success Guarantee	User name, username, password, and email are saved. If the client logs out, they will be able to log back in.
Main Success Scenario	1. User runs application, bringing them to the home screen. 2. User clicks on the "Login or Register" button, bringing them to the login screen. 3. User enters name, email address, password, and email. 4. User reads terms of agreement and checks the checkbox. 5. User clicks the finish registration button. They are then brought to the confirmation screen. 6. A confirmation key is sent to the email that the user registered with. 7. User retrieves key from email and types it into the confirmation field. User then clicks the Verify button. 8. A message is displayed informing the user that they successfully registered their account.
Extensions	3a. User does not completely enter in required information before hitting register. 1. System displays message telling the user to fill out the remaining fields.

	3b. User enters a non-UNH email. 1. System displays message telling the user that the email must be a valid UNH email. 3c. Email chosen already exists in the database. 1. Systems displays message to the user requesting the user to change their email. 5a. User does not check the registration checkbox. 1. System displays message telling the user that they must accept the terms of agreement before proceeding with registration. 7a. User enters invalid registration key. 1. System displays message telling the user that the verification key was invalid. System also informs user to check their junk mail, as email settings could cause the email to be received by the user. User is not verified in the database.
Special Requirements	N/A
Technology and Data Variations	Using a remote MySQL server to handle multiple users and store user information.
Frequency of Occurrence	Could be nearly continuous.
Miscellaneous	N/A

2. Sell Book

Use Case Section	Description
Use Case Name	Sell Book
Scope	*University of New Hampshire Textbook Exchange*
Level	User-Goal
Primary Actor	User
Stakeholders and Interests	User: Wants to upload a book to the database so it can be sold. Server: Needs to store information to the client.
Preconditions	User has ISBN number, book condition, and book price. User is logged in and validated. ISBN must be valid.
Success Guarantee	All of the book information is saved to the database. If someone searches for the book uploaded, it will show up as available to buy.
Main Success Scenario	1. User clicks on the "My Account" button, if not there already. 2. User clicks "Sell Book" button. User is brought to a page where book information can be entered. 3. User enters ISBN number, book condition, and book price. Book condition is chosen from pre-determined values within a drop down menu. 4. User clicks "Preview Sale" button. 5. The book information, which includes the book image and attributes, is displayed below in the format that it would appear to another user wanting to buy the book. 6. User clicks the "Sell this Book!" button to finalize sale and add book into the database.
Extensions	3a. User does not enter a valid ISBN number (Must be either an ISBN number or an ISBN13). 1. System displays a message informing the user that the ISBN entered was not valid. User is not allowed to proceed unless the ISBN entered is valid.

	3b. User does not enter a valid currency in the price field. 1. System displays a message informing the user that the price entered was not valid. User is not allowed to proceed unless the price entered is valid. 3c. User does not completely enter in required information before hitting register. 1. System displays message telling the client that all fields must be filled in.
Special Requirements	N/A
Technology and Data Variations	Using a remote MySQL server to store book information.
Frequency of Occurrence	User Dependent.
Miscellaneous	N/A

3. Buy Book

Use Case Section	Description
Use Case Name	Search Book
Scope	*University of New Hampshire Textbook Exchange*
Level	User-Goal
Primary Actor	User
Stakeholders and Interests	User: Wants to buy a book listed in the database by another user. Server: Needs to distribute information to the client.
Preconditions	User is logged in and validated.
Success Guarantee	Books will be displayed based on search category and value.
Main Success Scenario	1. User clicks the "My Account" button, if not there already. 2. User selects a category using a drop down menu, whose predetermined values are corresponding columns of the "Book Information" table in the MySQL database. 3. User enters text in the search field representing their desired information. 4. Table is displayed based on search results. 5. User clicks the table once and the book picture and book information is displayed above the table. A "Buy this Book!" button is also displayed. 6. User clicks on the "Buy this Book!" the button. 7. System informing the user that an email will be sent to the seller of the book clicked on. If desired, the user is given an option to add a personal message to the seller. 8. When O.K. is clicked, the email is sent and the user can continue proceed with the application.
Extensions	5a. User rapidly clicks between tables, causing multiple requests through the Amazon API in a short amount of time.

	1. Information from server requests is stored in a hash table. Book information then does not need to be fetched multiple times on different table clicks. 7a. User tries to send an email to a book that the user themselves had already listed. 1. System displays a message informing the user that they cannot buy their own book. 7b. User tried to send an email to a seller when the user already sent an email to that specific seller about the book in question. 1. System displays a message informing the user that an email was already sent to the seller about the book in question. It also informs the user that the user email was provided to the seller and the seller would contact the user if there was interest in selling the book.
Special Requirements	N/A
Technology and Data Variations	Using a remote MySQL server retrieve book information.
Frequency of Occurrence	User Dependent.
Miscellaneous	N/A

4. Update Account

Use Case Section	Description
Use Case Name	Edit User
Scope	*University of New Hampshire Textbook Exchange*
Level	User-Goal
Primary Actor	Administrator
Stakeholders and Interests	User: Wants to update user information. Server: Need to respond to user requests for modifications.
Preconditions	User is logged in and validated.
Success Guarantee	Server correctly responds to user requests for modification. No other fields besides the user's request are modified or deleted.
Main Success Scenario	1. User clicks the "My Account" button, if not there already. 2. User clicks on the "Modify Information" button. A screen is displayed showing the users current information. 3. User is able to modify the text fields corresponding to their information stored in the server by simply clicking within them and typing the value to be modified to. 4. User clicks on the "Accept Changes" button. Information is successfully modified within the server in accordance to the user request.
Extensions	3a. User does not completely enter in required information before hitting register. 1. System displays message telling the user to fill out the remaining fields. 3b. User enters a non-UNH email. 1. System displays message telling the user that the email must be a valid UNH email. 3c. User does not completely enter in required information

	before hitting register. 1. System displays message telling the client that all fields must be filled in.
Special Requirements	N/A
Technology and Data Variations	Using a remote MySQL server to update the information in the database.
Frequency of Occurrence	User dependent.
Miscellaneous	N/A

5. Manage User

Use Case Section	Description
Use Case Name	Manage User
Scope	*University of New Hampshire Textbook Exchange*
Level	User-Goal
Primary Actor	Administrator
Stakeholders and Interests	User: Wants to modify or delete user and book information. Server: Need to respond to client requests for modifications and deletions.
Preconditions	User must be an administrator.
Success Guarantee	Server correctly responds to client request for deletion and modification. No other fields besides the client's request are modified or deleted.
Main Success Scenario	1. Administrator logs in. 2. Administrator clicks on "Manage" button. 3. Administrator clicks on "Display Users" or "Display Books". 4. A table displaying all users is displayed. 5. Administrator double clicks on an entry. 6. Administrator chooses either "Delete" or "Modify" a. If Administrator chooses "Delete", a warning message is displayed, confirming that the administrator wants to delete the user. Administrator types "DELETE" to confirm. b. If Administrator chooses "Modify", a drop down menu displaying possible modification choices is displayed. User chooses a modification field. Current value of that field is displayed. Administrator types in the desired new value, and clicks OK.

Extensions	6a. Administrator does not enter "DELETE" correctly to confirm. 1. Database is not updated and user is allowed to re-enter the deletion process. 6b. Administrator tries to delete or modify another administrator's information. 1. System displays a message informing the administrator that other administrators cannot be modified. The database is not updated.
Special Requirements	N/A
Technology and Data Variations	Using a remote MySQL server to update the information in the database.
Frequency of Occurrence	Administrator dependent.
Miscellaneous	N/A

6 – Edit Book

Use Case Section	Description
Use Case Name	Edit Book
Scope	*University of New Hampshire Textbook Exchange*
Level	User-Goal
Primary Actor	User
Stakeholders and Interests	User: Wants to update book information. Server: Need to respond to user requests for modifications.
Preconditions	User is logged in and validated.
Success Guarantee	Server correctly responds to user requests for modification. No other fields besides the user's request are modified or deleted.
Main Success Scenario	1. User clicks the "My Account" button, if not there already. 2. User clicks on the "Manage Books" button. A screen is displayed showing the users current information. 3. User double clicks on a book record. 4. User chooses either "Delete" or "Modify" a. If User chooses "Delete", a warning message is displayed, confirming that the user wants to delete the book. User types "DELETE" to confirm. 5. If User chooses "Modify", a drop down menu displaying possible modification choices is displayed. User chooses a modification field. Current value of that field is displayed. User types in the desired new value, and clicks OK. 6. User clicks on the "Accept Changes" button. Information is successfully modified within the server in accordance to the user request.
Extensions	4a. User does not enter "DELETE" correctly to confirm.

	1. Database is not updated and user is allowed to re-enter the deletion process. 5a. User enters an invalid price. 2. System displays message telling the user that the price must be in valid U.S. currency format, and provides an example.
Special Requirements	N/A
Technology and Data Variations	Using a remote MySQL server to update the information in the database.
Frequency of Occurrence	User dependent.
Miscellaneous	N/A

7. Customize Page

Use Case Section	Description
Use Case Name	Customize Page
Scope	*University of New Hampshire Textbook Exchange*
Level	User-Goal
Primary Actor	User
Stakeholders and Interests	User: Wants to customize the look and feel of the application in order to achieve aesthetic satisfaction. Server: Needs to store user customization. When logging back in, user loads the previous customization state from the server.
Preconditions	User must be logged in.
Success Guarantee	User customization is successfully stored in the database. Customization state is saved on logout.
Main Success Scenario	1. User logs in. 2. User clicks on customize button. 3. User double clicks on highlighting color, even row color, and odd row color textboxes. 4. User selects a color from the color dialog. 5. User double clicks on header font and table font textboxes. 6. User selects a font from the font dialog. 7. User clicks the “Accept Changes” button.
Extensions	*a. At any time, User changes a customization value. 1 A sample data table is updated based on the users changed customization value. *b. At any time, User dislikes changes made and wants to revert to the default color and font scheme.

	1 User clicks on the “Revert to Default” button. User then clicks on the “Accept Changes” button to save default customization to the database.
Special Requirements	N/A
Technology and Data Variations	Using a remote MySQL server to store user customizations.
Frequency of Occurrence	User dependent.
Miscellaneous	N/A

8. View Transactions

Use Case Section	Description
Use Case Name	View Transactions
Scope	*University of New Hampshire Textbook Exchange*
Level	User-Goal
Primary Actor	User
Stakeholders and Interests	User: Wants to be able to view the books that they requested and the requests of the books that they've sold. Server: Needs to distribute transaction information.
Preconditions	User must be logged in.
Success Guarantee	All transactions associated with the user and the user's sold books are displayed. No information in the database will be altered.
Main Success Scenario	1. User clicks on the "My Account" button, if not there already. 2. User clicks on the "Transaction History" button. 3. User can then click on the "Sales Pending" or the "Purchases Pending" pane to view the transaction information.
Extensions	N/A
Special Requirements	N/A
Technology and Data Variations	Using a remote MySQL server to store user customizations.
Frequency of Occurrence	User dependent.
Miscellaneous	N/A

2.2. Software Qualities

This application focuses heavily on providing a user friendly environment. The main goals while creating the environment were to provide an intuitive, modern, robust, reactive, and aesthetically pleasing interface. This was done through careful font, color, and Look and Feel choice and of the user interface, through ensuring the application scales according to the frame size and the user's screen resolution, and through providing the user with feedback for every action they perform in the interface.

The application also gives the user the ability to customize the table used for displaying book information by changing the colors and fonts of their environment. This information is stored in a relational database table on *MySql* and is reloaded when the user logs in. By allowing the user to manipulate the table information, the application has also become accessible to those with visual impairment. This includes users who are colorblind or have poor eye sight. The application is also scalable based on the frame size. All components of the user interface will automatically adjust their location depending on the user's desired frame size. If the user makes the frame size too small to display all of the components, a horizontal or vertical scroll bar is inserted as needed. Examples of the intuitive, modern, and aesthetically pleasing interface can be seen in Section 5, User Documentation.

The application is also robust, ensuring the user does not encounter unexpected behavior when using the application. Every time the application processes user input the input is brought through a series of tests that ensure the data entered is valid and conforms to the information requested in that specific field. For example, only real ISBN numbers can be entered in the sell book page and only e-mail addresses with a UNH extension can be used for an account's e-mail. Other examples of the robustness of the program include ensuring the user only has access to their own information even after multiple users have logged in and out of the application, all fields are updated with the most current user information at all times (such as custom information, book information, transaction history, etc.), and through ensuring e-mails are properly sent to the user during the registration process,

password retrieval process, and book purchasing process. Through thorough black box testing, white box testing, and runtime error checks within the program the user experience has been proven to be reliable. These tests and error checks are discussed in more detail in Section 6, Software Documentation.

Lastly, this application is extremely resource efficient. The user spends minimal time waiting for database and Amazon queries. This ensures the user satisfaction because the user is able to quickly navigate between pages, update their sold book list, and browse through up for sale books. Resource efficiency is discussed in more detail in Section 4, Technical Documentation.

2.3. System Constraints

By choosing *Java* as the programming language for the *University of New Hampshire Textbook Exchange*, the application is able to run on all operating systems with a *Java Virtual Machine*. By making our application accessible to *Linux*, *MAC*, and *Windows* users, the application is able to hit nearly all of the target audience. One requirement for using this application is a valid internet connection. This is necessary in order to access the *MySQL* server, request products from *Amazon*, and to send e-mails using the *Secure Mail Transfer Protocol (SMTP)* server.

3. Design Specifications

3.1. Software Architecture

3.1.1. Object Model

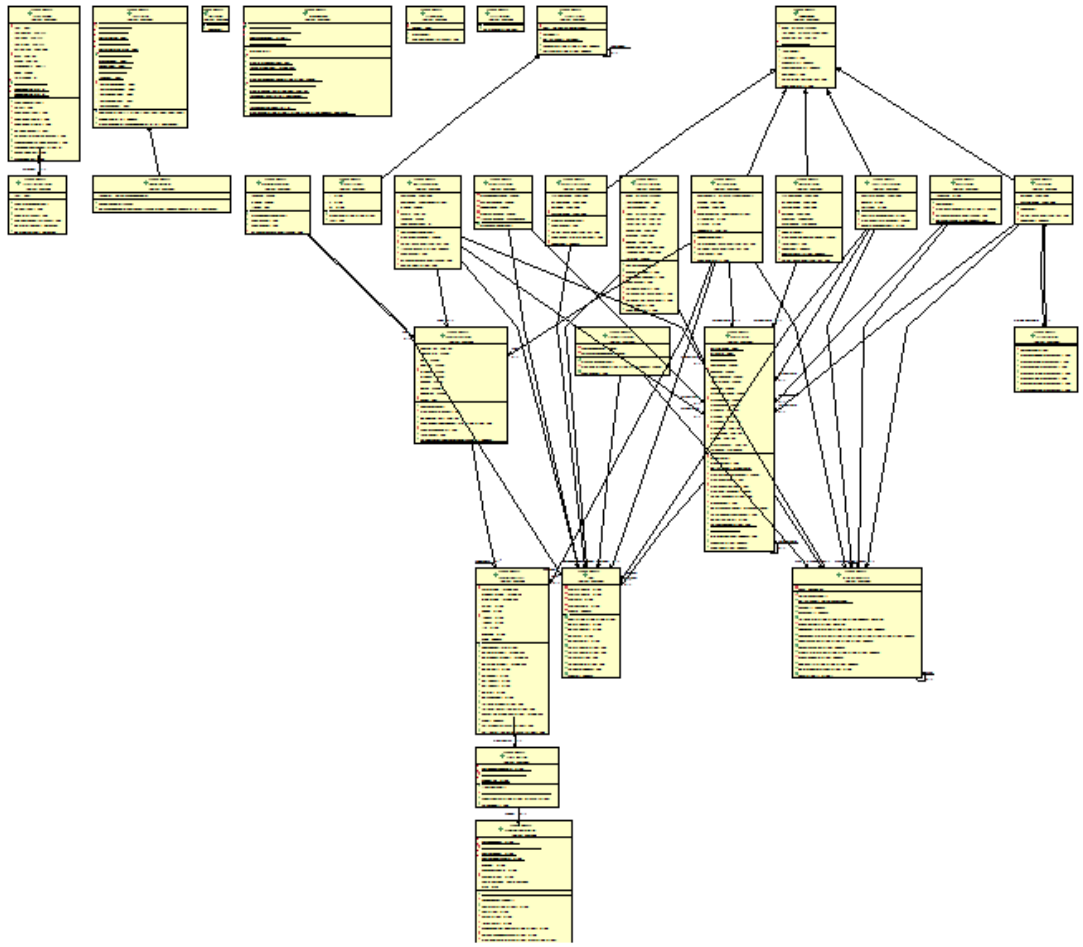


Figure 3.1.1.1 University of New Hampshire Textbook Exchange UML Diagram

This is the class diagram for the all of the classes in the University of New Hampshire Textbook Exchange.

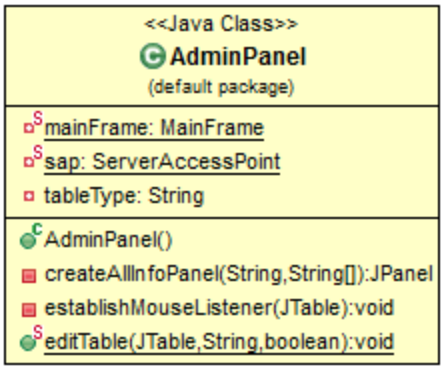


Figure 3.1.1.2AdminPanel UML Diagram

This is the Admin Panel UML Diagram. The instance variables are listed underneath the class name and the methods are below the instance variables. To see a larger picture, reference Figure 3.1.1.1.

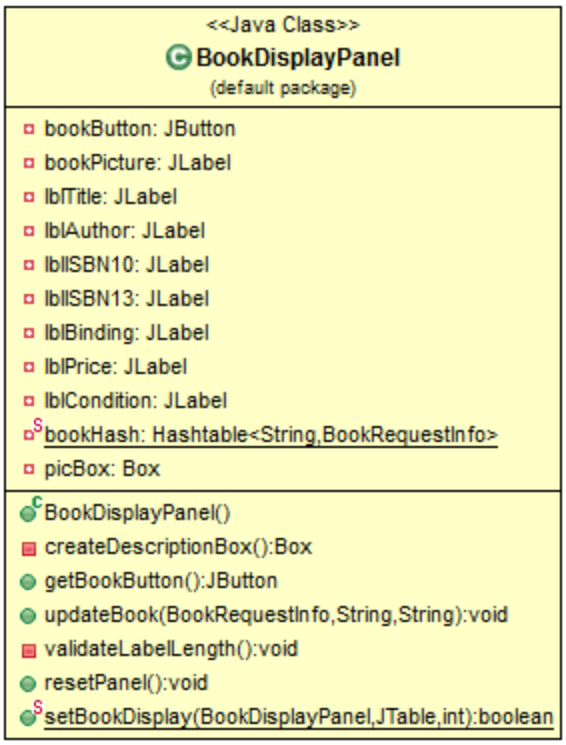


Figure 3.1.1.3 Book Display Panel UML Diagram

This is the Book Display Panel UML Diagram. The instance variables are listed underneath the class name and the methods are below the instance variables. To see a larger picture, reference Figure 3.1.1.1.

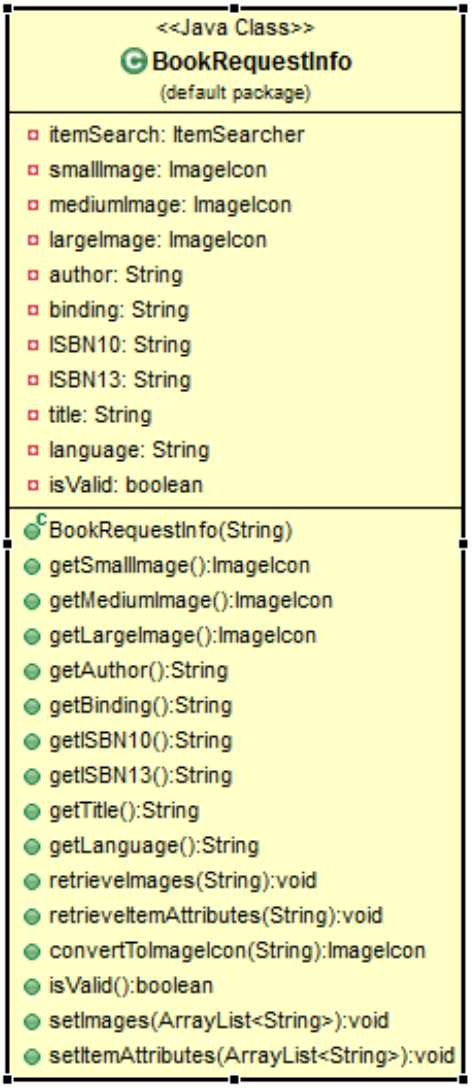


Figure 3.1.1.4 Book Request Info UML Diagram

This is the Book Request Info UML Diagram. The instance variables are listed underneath the class name and the methods are below the instance variables. To see a larger picture, reference Figure 3.1.1.1.

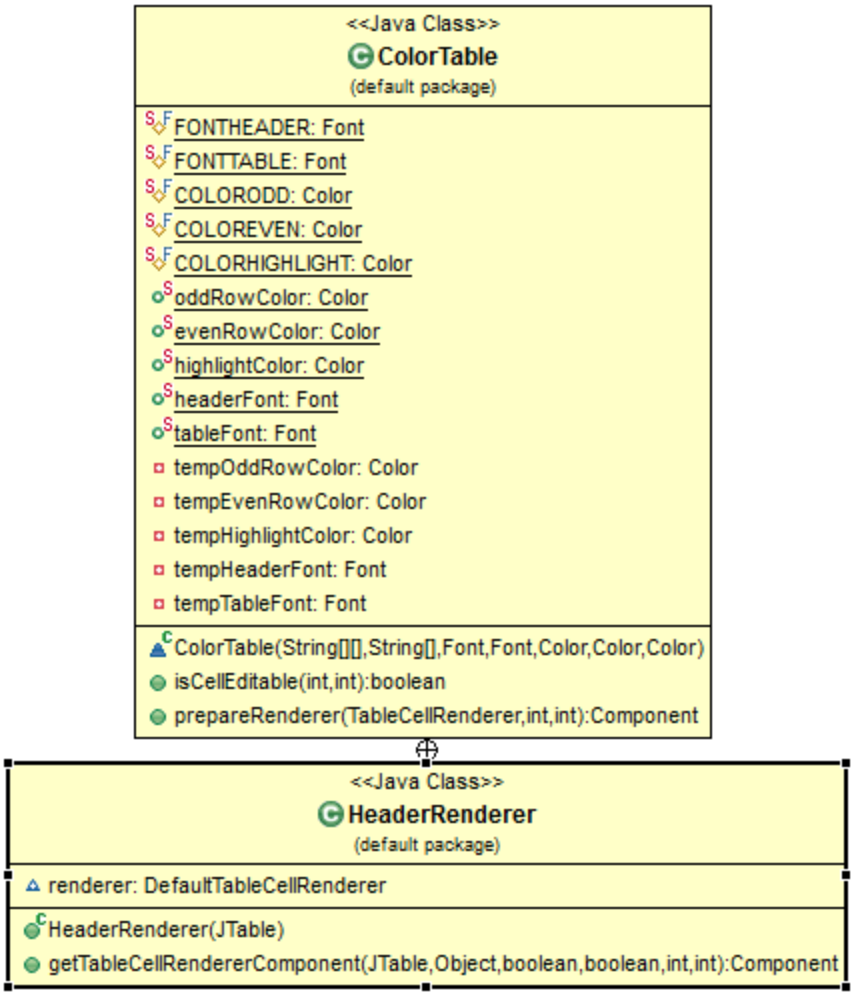


Figure 3.1.1.5 Color Table UML Diagram

This is the Color Table UML Diagram. The instance variables are listed underneath the class name and the methods are below the instance variables. To see a larger picture, reference Figure 3.1.1.1.

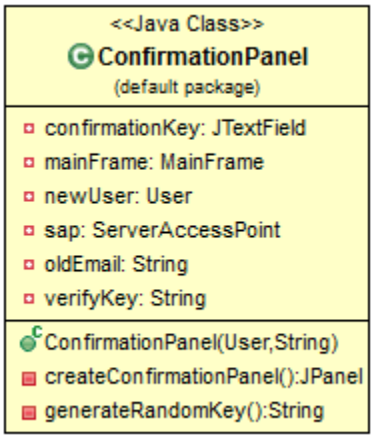


Figure 3.1.1.6 Confirmation Panel UML Diagram

This is the Confirmation Panel UML Diagram. The instance variables are listed underneath the class name and the methods are below the instance variables. To see a larger picture, reference Figure 3.1.1.1.

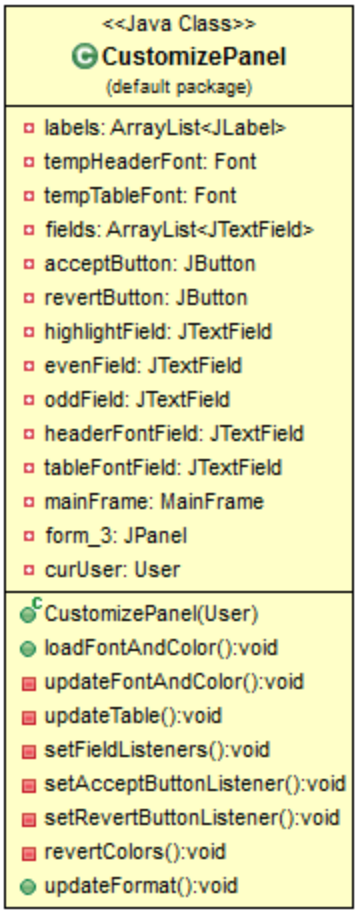


Figure 3.1.1.7 Customize Panel UML Diagram

This is the Customize Panel UML Diagram. The instance variables are listed underneath the class name and the methods are below the instance variables. To see a larger picture, reference Figure 3.1.1.1.

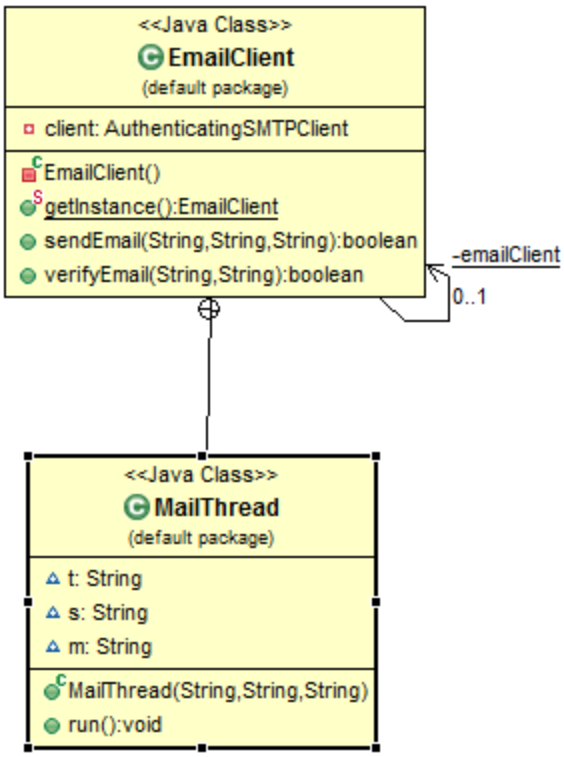


Figure 3.1.1.8 Email Client UML Diagram

This is the Email Client Panel UML Diagram. The instance variables are listed underneath the class name and the methods are below the instance variables. To see a larger picture, reference Figure 3.1.1.1.

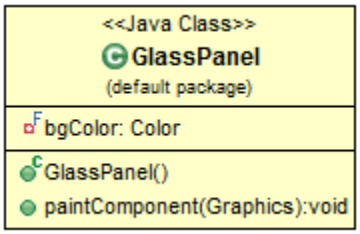


Figure 3.1.1.9 Glass Panel UML Diagram

This is the Glass Panel UML Diagram. The instance variables are listed underneath the class name and the methods are below the instance variables. To see a larger picture, reference Figure 3.1.1.1.

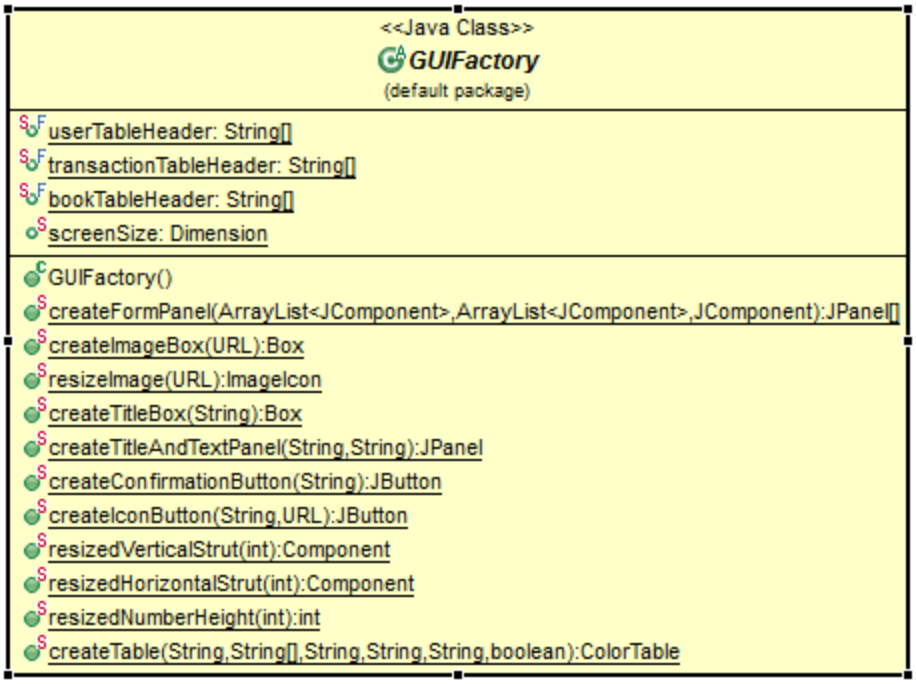


Figure 3.1.1.10 GUI Factory UML Diagram

This is the GUI Factory UML Diagram. The instance variables are listed underneath the class name and the methods are below the instance variables. To see a larger picture, reference Figure 3.1.1.1.

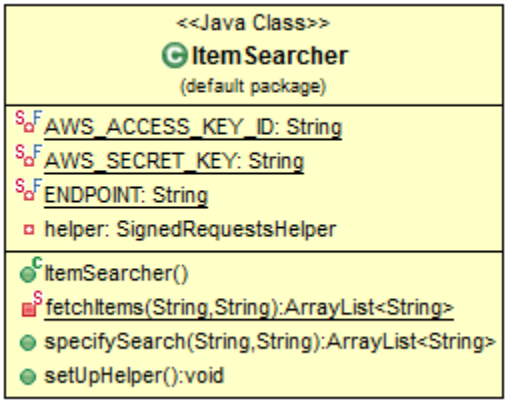


Figure 3.1.1.11 Item Searcher UML Diagram

This is the Item Searcher UML Diagram. The instance variables are listed underneath the class name and the methods are below the instance variables. To see a larger picture, reference Figure 3.1.1.1.

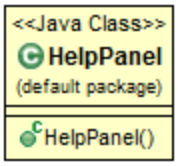


Figure 3.1.1.12 Help Panel UML Diagram

This is the Help Panel UML Diagram. The methods are below the instance variables. To see a larger picture, reference Figure 3.1.1.1.

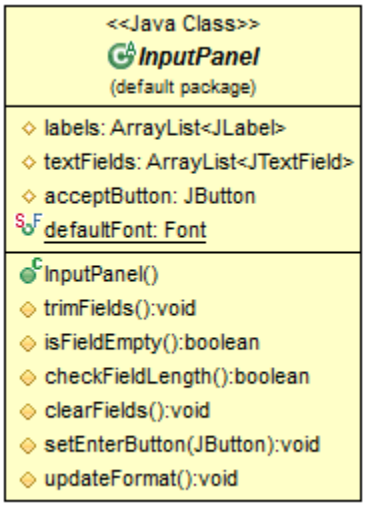


Figure 3.1.1.12AdminPanel UML Diagram

This is the Input Panel UML Diagram. The instance variables are listed underneath the class name and the methods are below the instance variables. To see a larger picture, reference Figure 3.1.1.1.

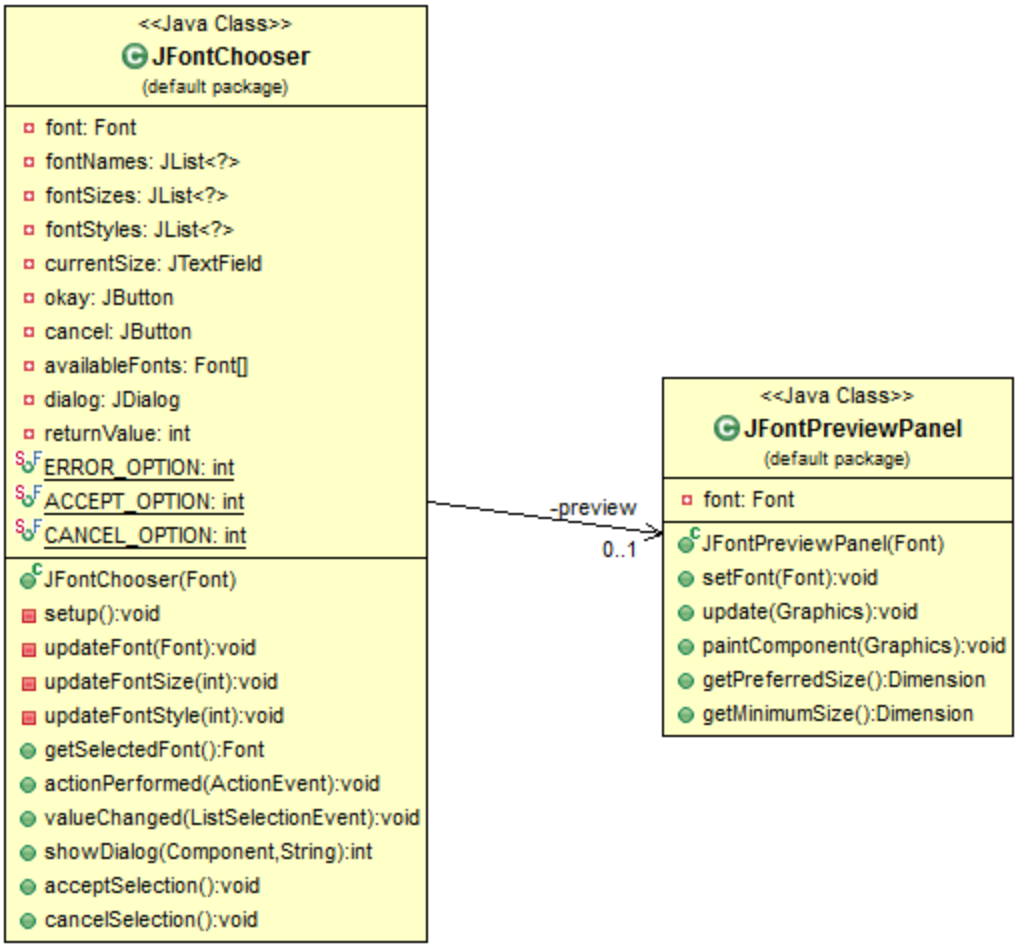


Figure 3.1.1.13 JFontChooser UML Diagram

This is the JFontChooser UML Diagram. The instance variables are listed underneath the class name and the methods are below the instance variables. To see a larger picture, reference Figure 3.1.1.1.

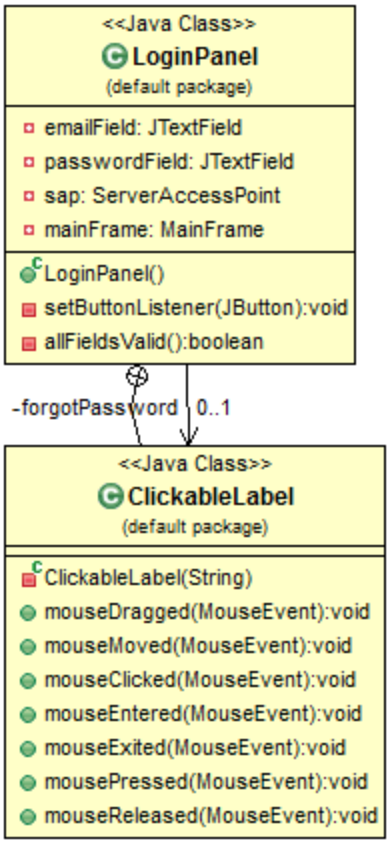


Figure 3.1.1.14 Login Panel UML Diagram

This is the Login Panel UML Diagram. The instance variables are listed underneath the class name and the methods are below the instance variables. To see a larger picture, reference Figure 3.1.1.1.

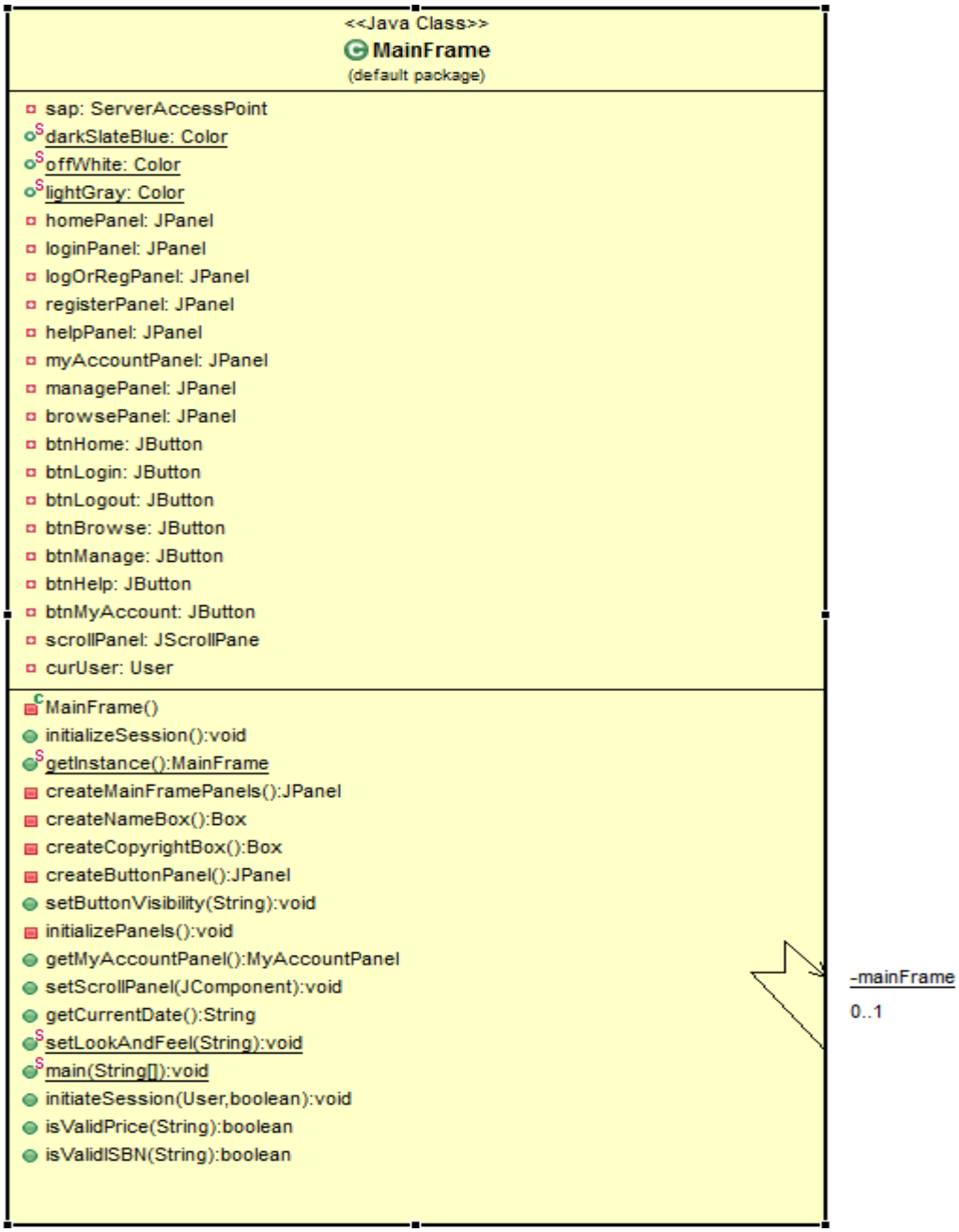


Figure 3.1.1.15 Mainframe UML Diagram

This is the Mainframe UML Diagram. The instance variables are listed underneath the class name and the methods are below the instance variables. To see a larger picture, reference Figure 3.1.1.1.

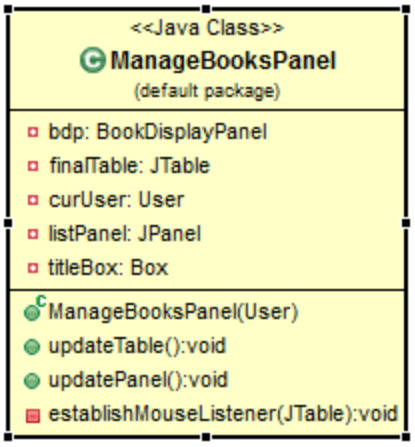


Figure 3.1.1.16 Mainframe UML Diagram

This is the Mainframe UML Diagram. The instance variables are listed underneath the class name and the methods are below the instance variables. To see a larger picture, reference Figure 3.1.1.1.

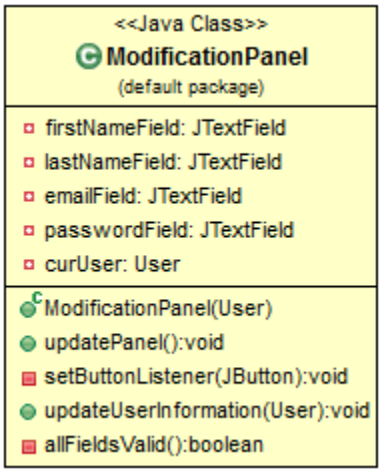


Figure 3.1.1.17 Modification Panel UML Diagram

This is the Modification Panel UML Diagram. The instance variables are listed underneath the class name and the methods are below the instance variables. To see a larger picture, reference Figure 3.1.1.1.

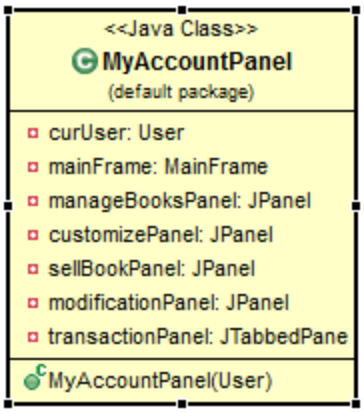


Figure 3.1.1.18 My Account Panel UML Diagram

This is the My Account Panel UML Diagram. The instance variables are listed underneath the class name and the methods are below the instance variables. To see a larger picture, reference Figure 3.1.1.1.

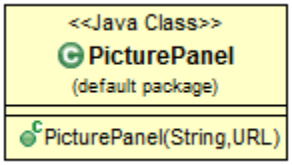


Figure 3.1.1.19 Picutre Panel UML Diagram

This is the Picture Panel UML Diagram. The instance variables are listed underneath the class name and the methods are below the instance variables. To see a larger picture, reference Figure 3.1.1.1.

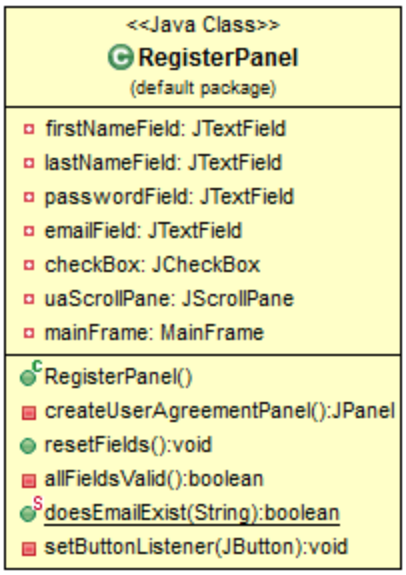


Figure 3.1.1.20 Register Panel UML Diagram

This is the Register Panel UML Diagram. The instance variables are listed underneath the class name and the methods are below the instance variables. To see a larger picture, reference Figure 3.1.1.1.

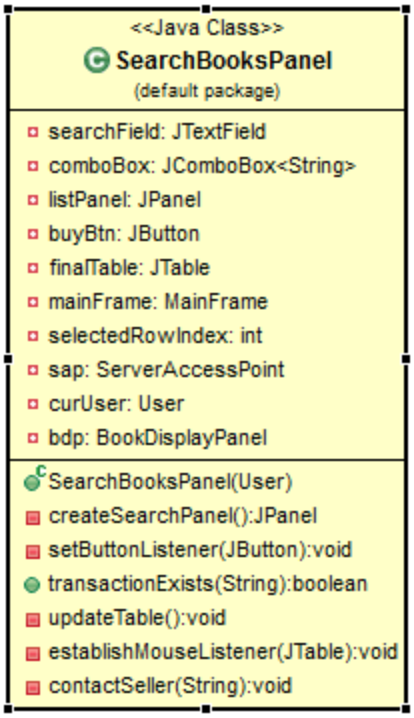


Figure 3.1.1.21 Search Books Panel UML Diagram

This is the Search Books Panel UML Diagram. The instance variables are listed underneath the class name and the methods are below the instance variables. To see a larger picture, reference Figure 3.1.1.1.

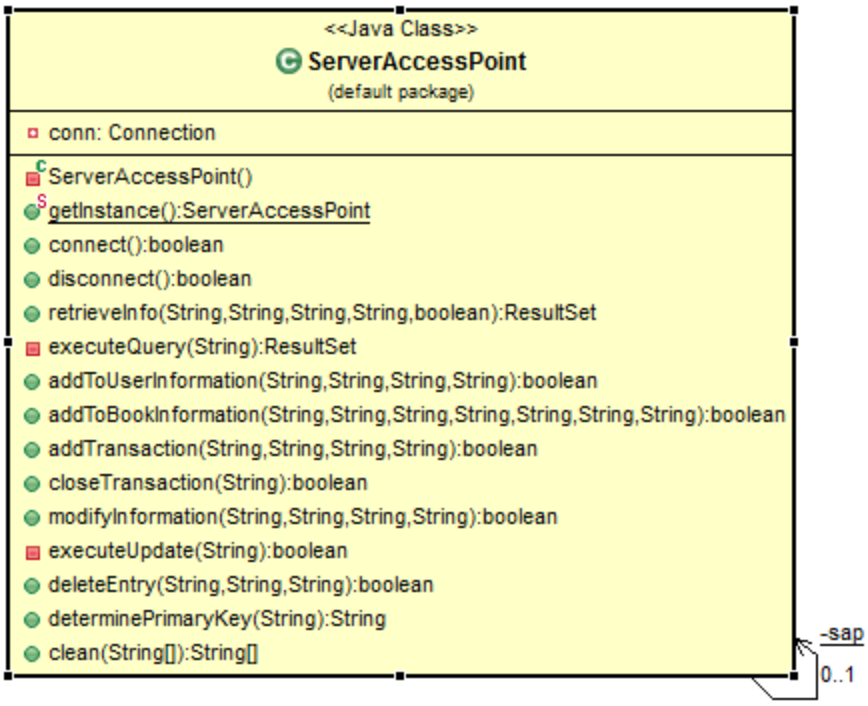


Figure 3.1.1.22 Search Books Panel UML Diagram

This is the Server Access Point UML Diagram. The instance variables are listed underneath the class name and the methods are below the instance variables. To see a larger picture, reference Figure 3.1.1.1.

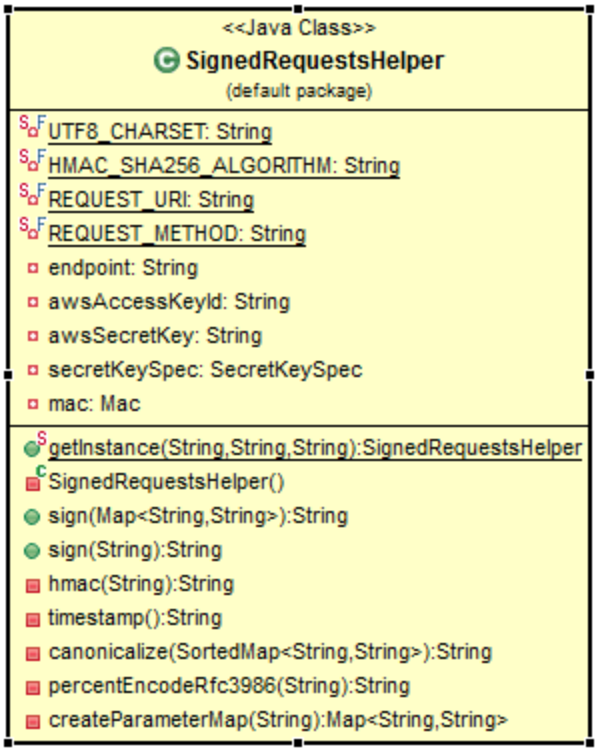


Figure 3.1.1.23 Signed Requests Helper UML Diagram

This is the Signed Requests Helper UML Diagram. The instance variables are listed underneath the class name and the methods are below the instance variables. To see a larger picture, reference Figure 3.1.1.1.

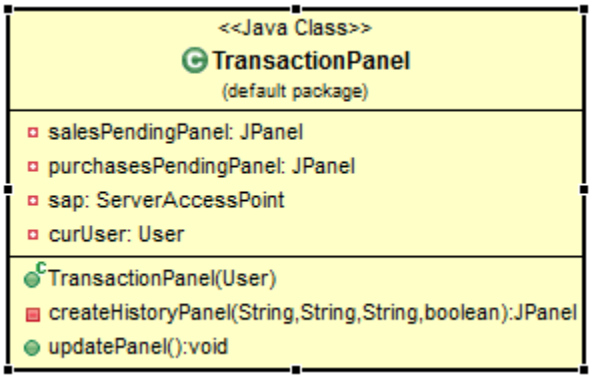


Figure 3.1.1.24 User UML Diagram

This is the User UML Diagram. The instance variables are listed underneath the class name and the methods are below the instance variables. To see a larger picture, reference Figure 3.1.1.1.

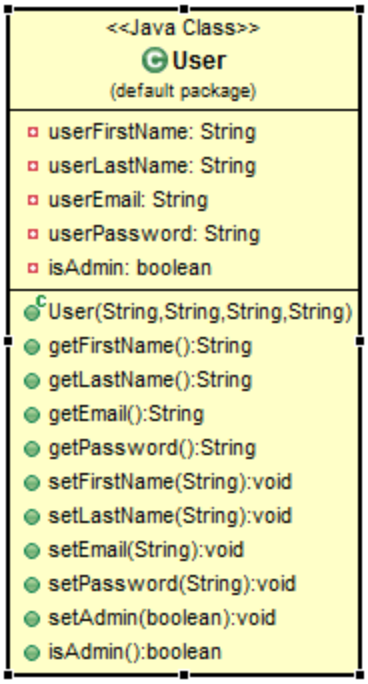


Figure 3.1.1.25AdminPanel UML Diagram

This is the Admin Panel UML Diagram. The instance variables are listed underneath the class name and the methods are below the instance variables. To see a larger picture, reference Figure 3.1.1.1.

3.1.2. User Interface Model

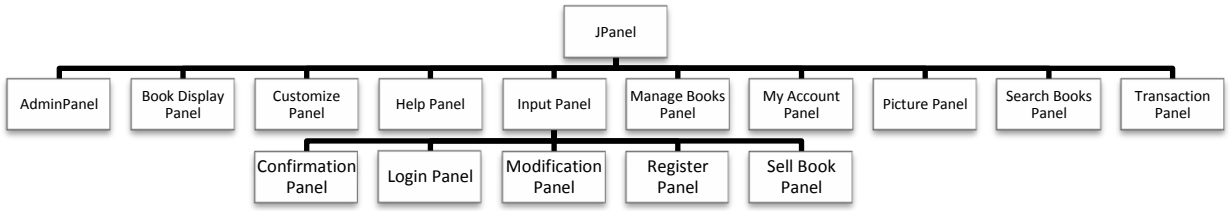


Figure 3.1.2.1 Panel Inheritence

The application Graphical User Interface swaps JPanels in and out of the main JFrame. Each class which inherits from JPanel is used as its own screen when interacting with the User Interface. The InputPanel class, however, is an abstract class, and as a result cannot be instantiated. The InputPanel class is used to encapsulate all panels that require the user to enter information. The InputPanel consists of many methods that make editing this input much simpler. It also provides convenience methods for validating user input.

3.1.3. Physical Data Model

For an overview of the physical data model go to Section 4.4, Database Tables

3.2. Sequence Diagrams

Use cases will be addressed in the following order:

1. Register User
2. Sell Book
3. Buy Book
4. Update Account
5. Manage User
6. Edit Book
7. Customize Page
8. View Transactions

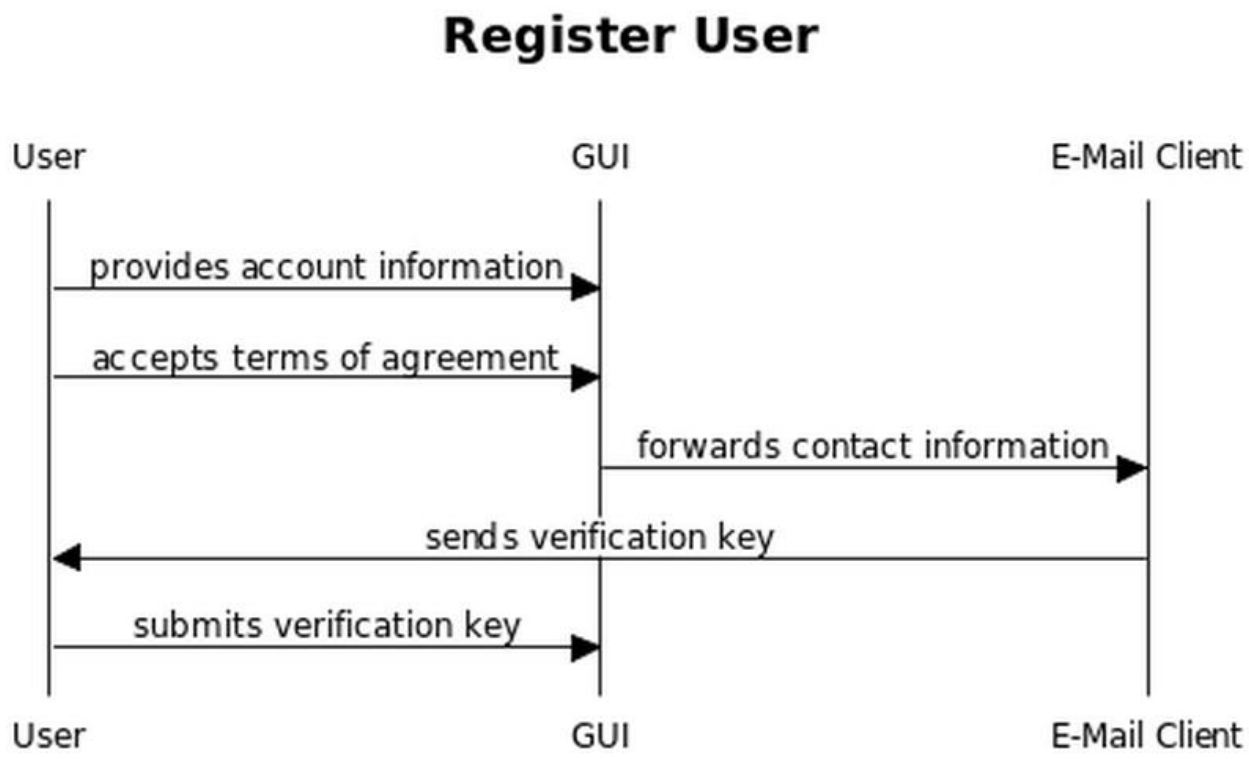


Figure 3.2.1: Register User Sequence Diagram

This is the sequence diagram associated with the Register User use case. See Section 2.1.2, Fully Dressed Use Cases for more detail.

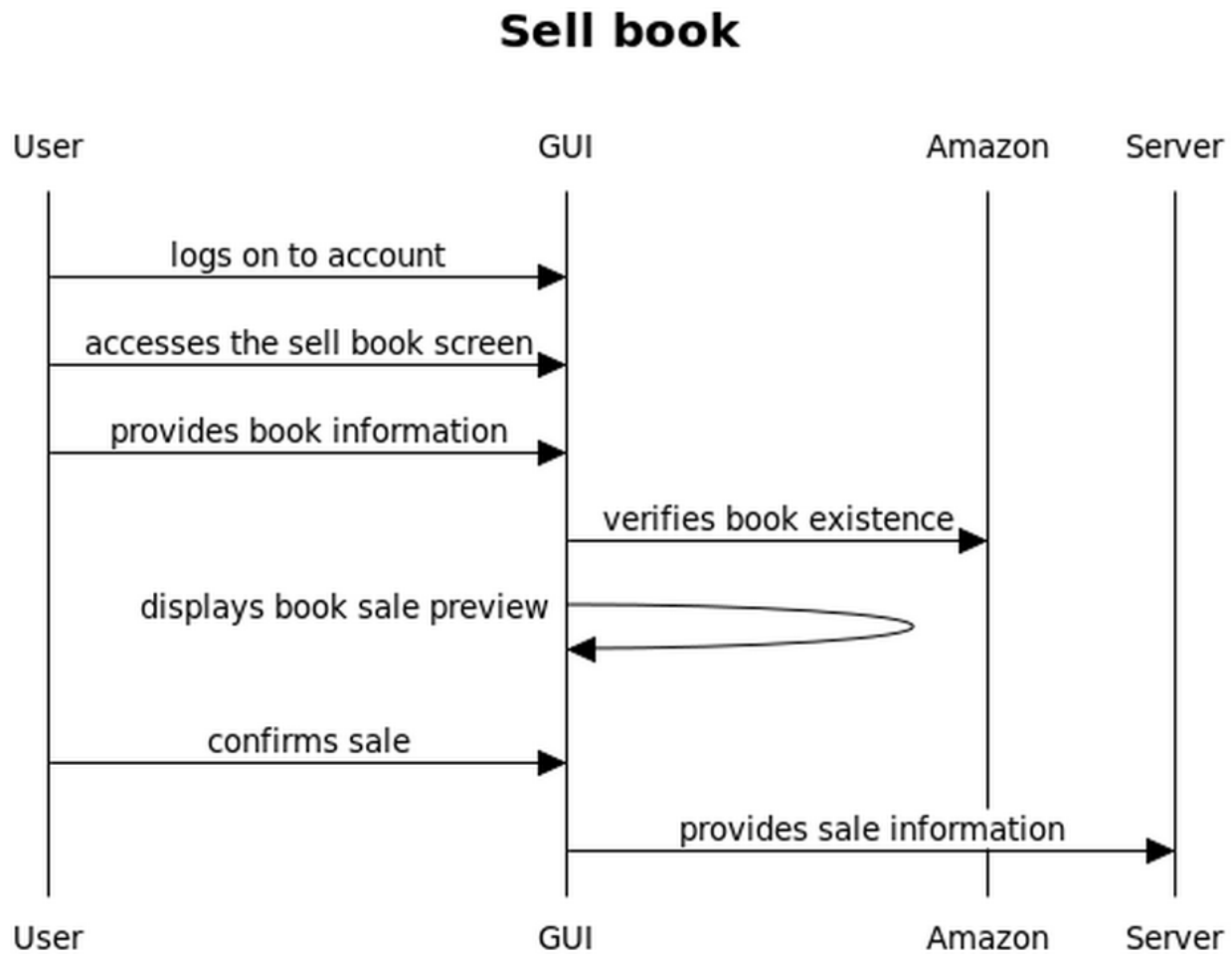


Figure 3.2.2: Sell Book Sequence Diagram

This is the sequence diagram associated with the Sell Book use case. See Section 2.1.2, Fully Dressed Use Cases for more detail.

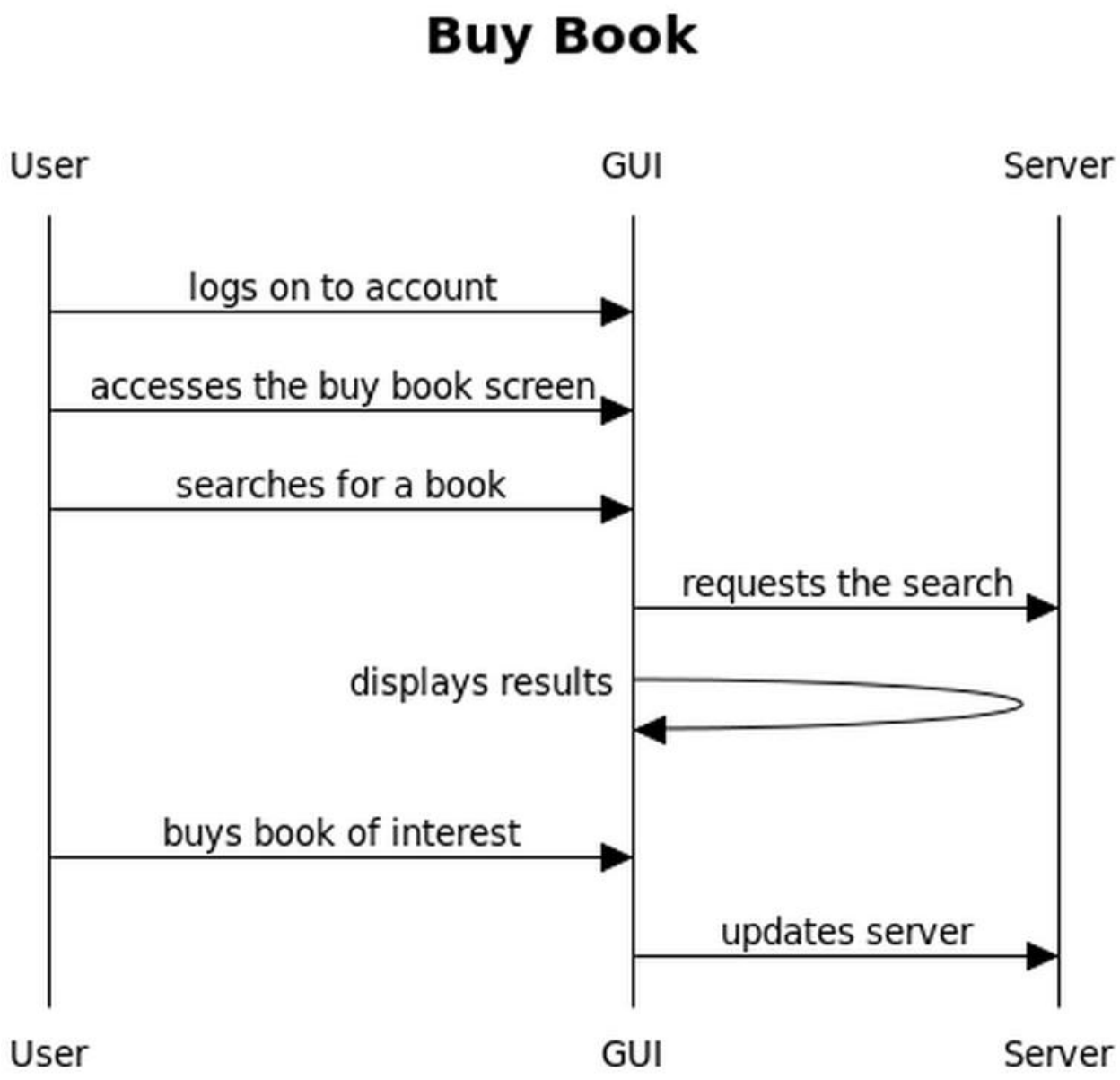


Figure 3.2.3: Buy Book Sequence Diagram

This is the sequence diagram associated with the Buy Book use case. See Section 2.1.2, Fully Dressed Use Cases for more detail.

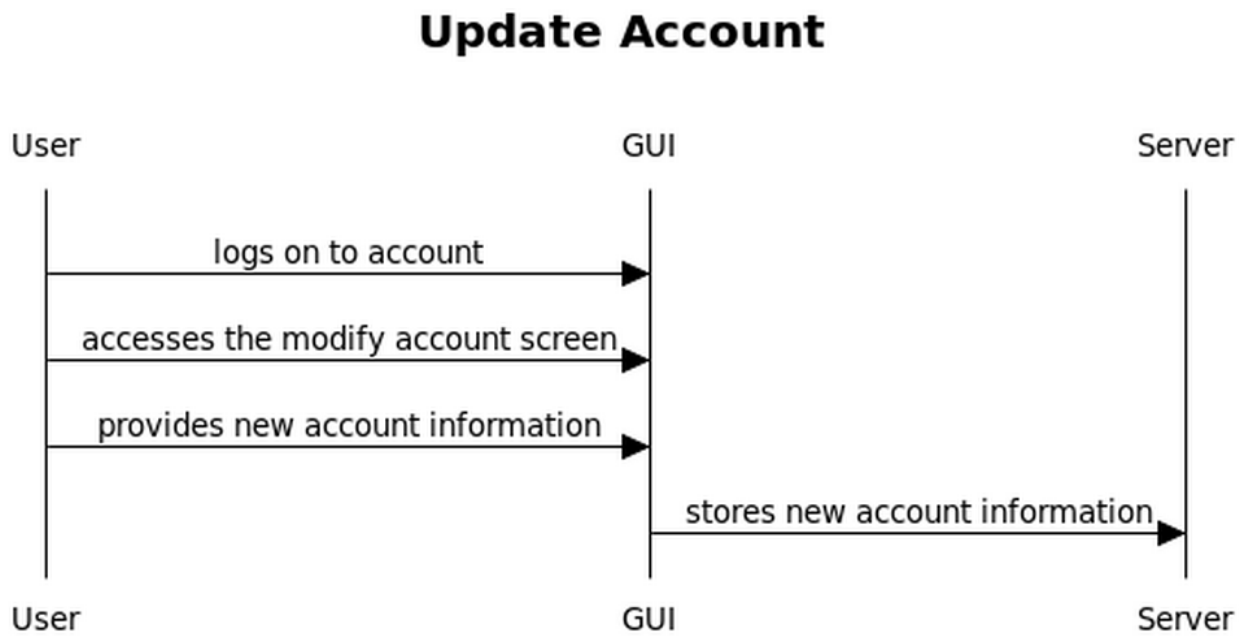


Figure 3.2.4: Update Account Sequence Diagram

This is the sequence diagram associated with the Update Account use case. See Section 2.1.2, Fully Dressed Use Cases for more detail.

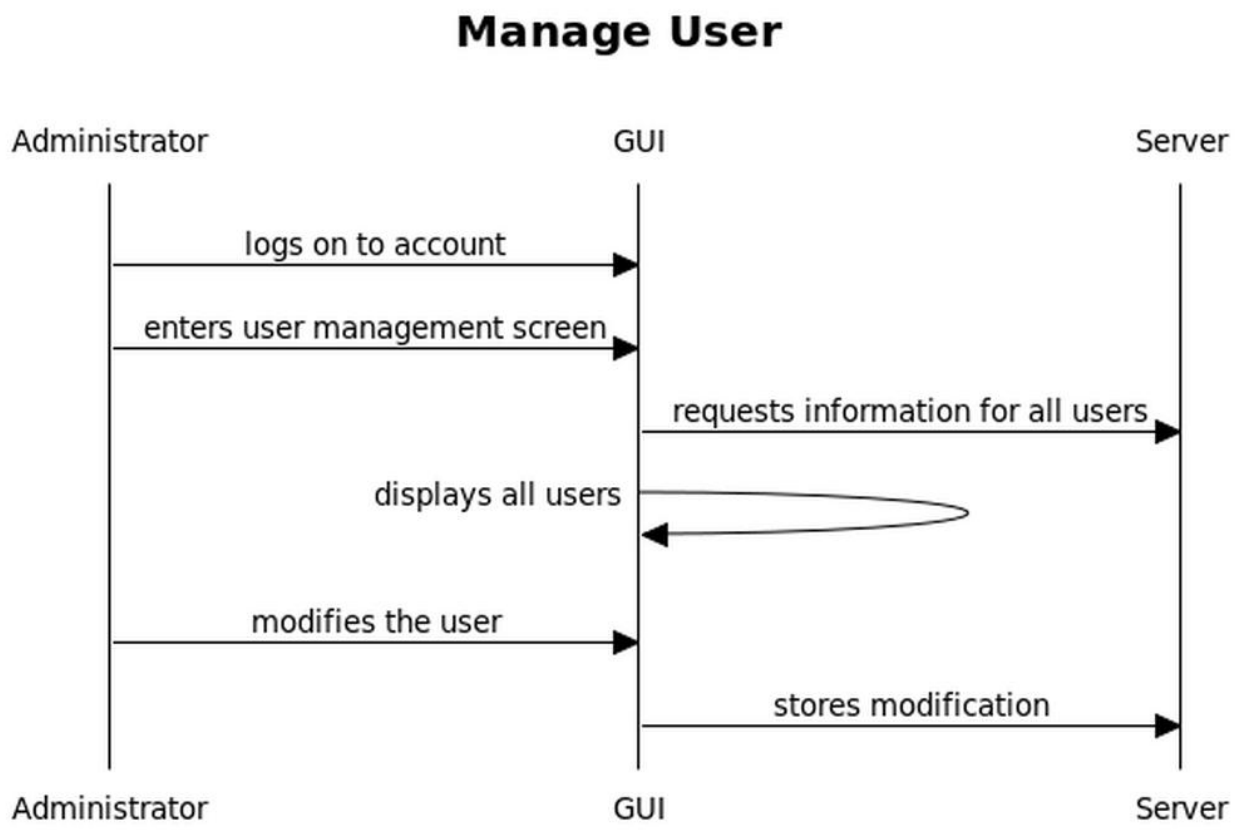


Figure 3.2.5: Manage User Sequence Diagram

This is the sequence diagram associated with the Manage User use case. See Section 2.1.2, Fully Dressed Use Cases for more detail.

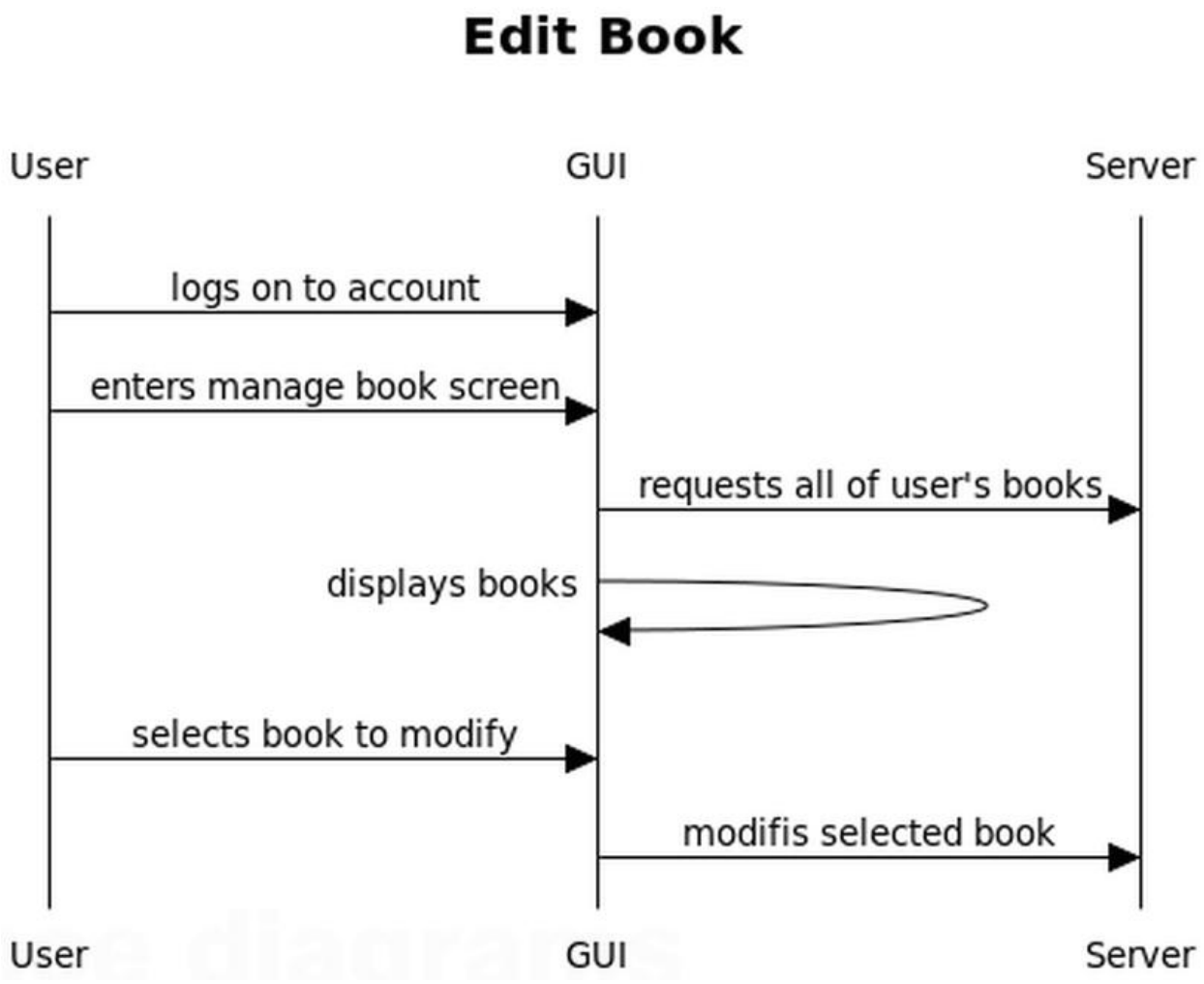


Figure 3.2.6: Edit Book Sequence Diagram

This is the sequence diagram associated with the Edit Book use case. See Section 2.1.2, Fully Dressed Use Cases for more detail.

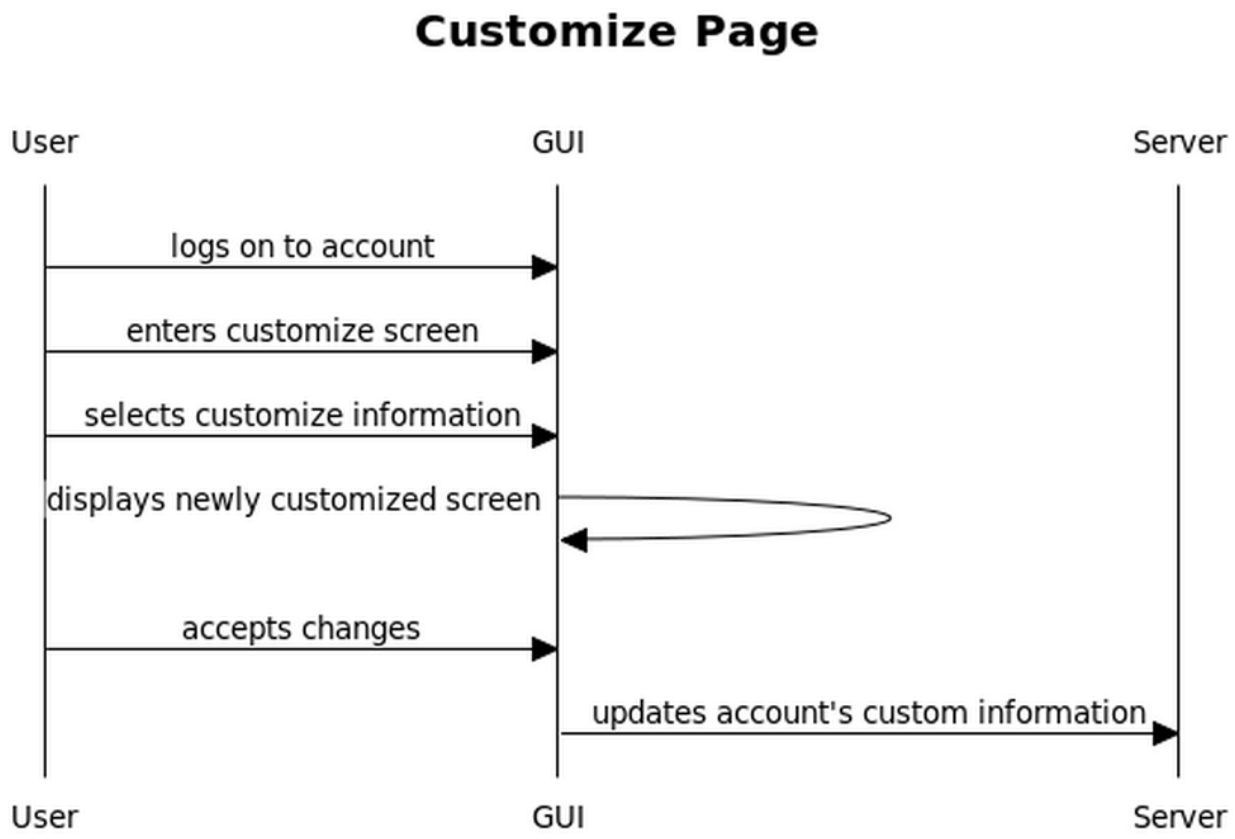


Figure 3.2.7: Customize Page Sequence Diagram

This is the sequence diagram associated with the Customize Page use case. See Section 2.1.2, Fully Dressed Use Cases for more detail.

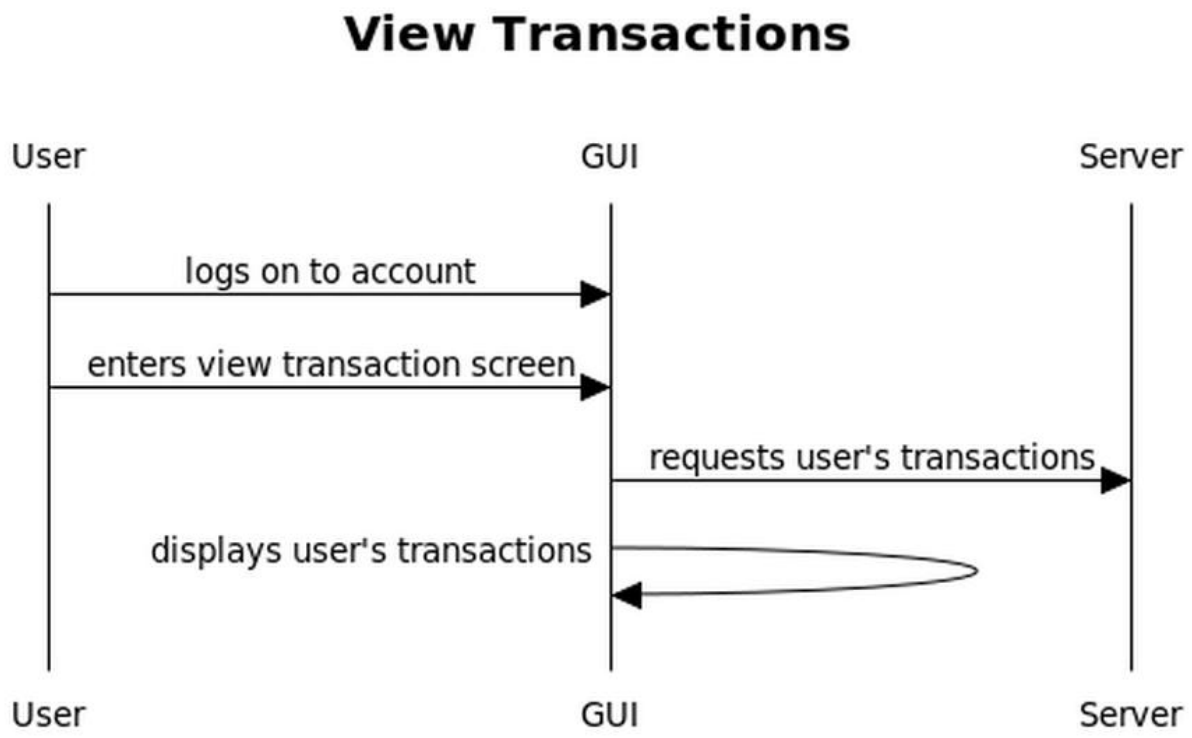


Figure 3.2.8: View Transactions Sequence Diagram

This is the sequence diagram associated with View Transactions use case. See Section 2.1.2, Fully Dressed Use Cases for more detail.

3.3. Class Diagrams

3.3.1. Domain Model

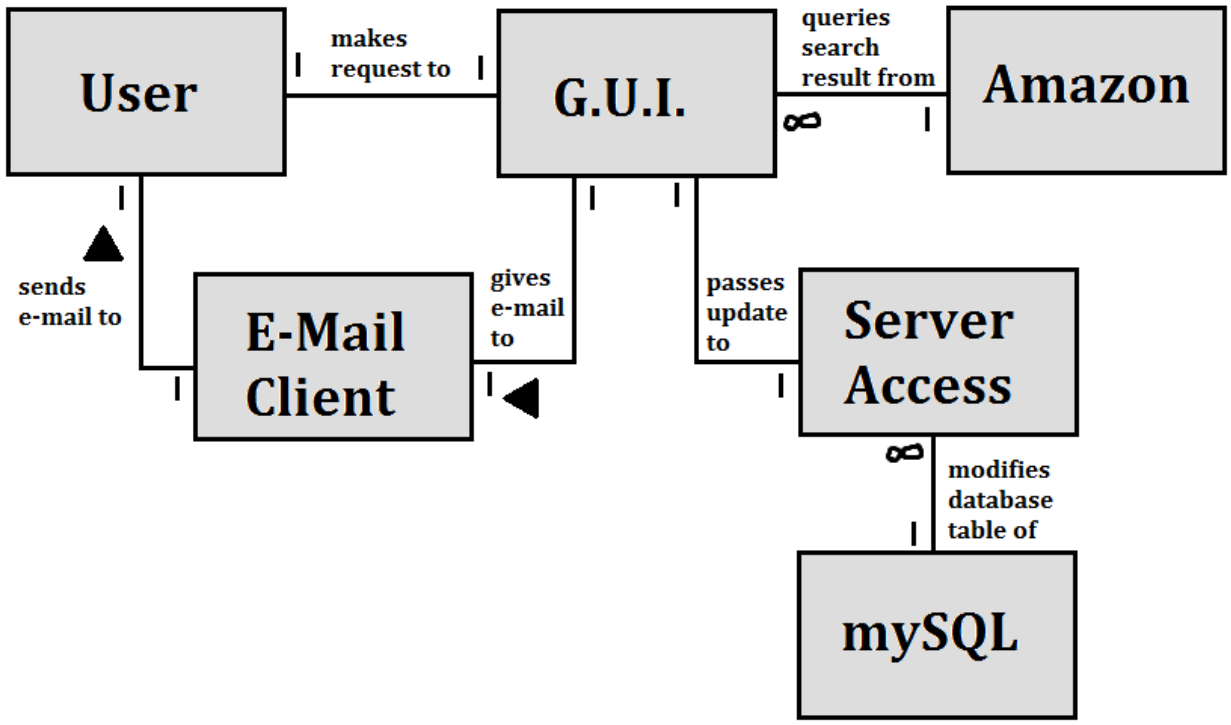


Figure 3.4.1.1

The domain model outlining the roles and relationships of the software model. There user is capable of making requests to the G.U.I who then forwards these requests to Amazon, the server access point, or the e-mail client where the request is handled accordingly. If the server access point receives the request the server access point generates a statement to modify the database in the SQL server.

3.3.2. Class Responsibility Collaboration Cards

The class responsibility collaboration (CRC) cards contain the class name at the top, responsibilities listed on the left, and collaborators listed on the right. The CRC cards are displayed in the following order:

1. AdminPanel
2. BookDisplayPanel
3. BookRequestInfo
4. ColorTable
5. ConfirmationPanel
6. CustomizePanel
7. EmailClient
8. GlassPanel
9. GUIFactory
10. HelpPanel
11. InputPanel
12. ItemSearcher
13. JFontChooser
14. LoginPanel
15. MainFrame
16. ManageBooksPanel
17. ModifiationPanel
18. MyAccountPanel
19. PicturePanel
20. RegisterPanel
21. SearchBooksPanel
22. SellBookPanel
23. ServerAccessPoint
24. SignedRequestsHelper
25. TransactionPanel
26. User

AdminPanel	
Delete books Delete users Modify users Update	ColorTable GlassPanel MainFrame ServerAccessPoint

Figure 3.3.1 AdminPanel CRC card

Class responsibilities are listed on the left and collaborators required to accomplish these responsibilities are listed on the right

BookDisplayPanel	
Display book photo Display book information (author ISBN etc.) ResetPanel	BookRequestInfo

Figure 3.3.2 BookDisplayPanel CRC card

Class responsibilities are listed on the left and collaborators required to accomplish these responsibilities are listed on the right

BookRequestInfo	
Create book photo Store book photo Store book information	ItemSearcher

Figure 3.3.3 BookRequestInfo CRC card

Class responsibilities are listed on the left and collaborators required to accomplish these responsibilities are listed on the right

ColorTable	
Create JTable Set color scheme Set font scheme	

Figure 3.3.4 ColorTable CRC card

Class responsibilities are listed on the left and collaborators required to accomplish these responsibilities are listed on the right

ConfirmationPanel	
Generate verification key Send verification email Validate verification key	EmailClient GlassPanel PicturePanel MainFrame

Figure 3.3.5 ConfirmationPanel CRC card

Class responsibilities are listed on the left and collaborators required to accomplish these responsibilities are listed on the right

CustomizePanel	
Revert to default color scheme Save color scheme Set color scheme Update preview table	ColorTable GlassPanel JFontChooser GUIFactory MainFrame PicturePanel SerrverAccessPoint

Figure 3.3.6 CustomizePanel CRC card

Class responsibilities are listed on the left and collaborators required to accomplish these responsibilities are listed on the right

EmailClient	
Send email	MailThread

Figure 3.3.7 EmailClient CRC card

Class responsibilities are listed on the left and collaborators required to accomplish these responsibilities are listed on the right

GlassPanel	
Fade screen	MainFrame

Figure 3.3.8 GlassPanel CRC card

Class responsibilities are listed on the left and collaborators required to accomplish these responsibilities are listed on the right

GUIFactory	
Create form Create table Create title box Resize Image	ServerAccessPoint

Figure 3.3.9 GUIFactory CRC card

Class responsibilities are listed on the left and collaborators required to accomplish these responsibilities are listed on the right

HelpPanel	
Display help information	GUIFactory MainFrame

Figure 3.3.10 HelpPanel CRC card

Class responsibilities are listed on the left and collaborators required to accomplish these responsibilities are listed on the right

InputPanel	
Clear fields Check field validity Set field listener to enter button Trim textfields	

Figure 3.3.11 InputPanel CRC card

Class responsibilities are listed on the left and collaborators required to accomplish these responsibilities are listed on the right

ItemSearcher	
Fetch items Specify search	SignedRequestsHelper

Figure 3.3.12 ItemSearcher CRC card

Class responsibilities are listed on the left and collaborators required to accomplish these responsibilities are listed on the right

JFontChooser	
Display font type Display font style Display font size Preview font	

Figure 3.3.13 JFontChooser CRC card

Class responsibilities are listed on the left and collaborators required to accomplish these responsibilities are listed on the right

LoginPanel	
Verify user information Retrieve password	EmailClient

Figure 3.3.14 LoginPanel CRC card

Class responsibilities are listed on the left and collaborators required to accomplish these responsibilities are listed on the right

MailThread	
Send email	EmailClient

Figure 3.3.15 MailThread CRC card

Class responsibilities are listed on the left and collaborators required to accomplish these responsibilities are listed on the right

MainFrame	
Determine button visibility Display base page Swap screens	AdminPanel GlassPanel HelpPanel Login Panel Picture Panel RegisterPanel ServerAccessPoint User

Figure 3.3.16 MainFrame CRC card

Class responsibilities are listed on the left and collaborators required to accomplish these responsibilities are listed on the right

ManageBooksPanel	
Modify book Update book table View book	BookDisplayPanel ColorTable GlassPanel GUIFactory MainFrame PictrePanel ServerAccessPoint User

Figure 3.3.17 ManageBooksPanel CRC card

Class responsibilities are listed on the left and collaborators required to accomplish these responsibilities are listed on the right

ModificationPanel	
Verify user information	GlassPanel GUIFactory MainFrame PicturePanel ServerAccessPoint User

Figure 3.3.18 ModificationPanel CRC card

Class responsibilities are listed on the left and collaborators required to accomplish these responsibilities are listed on the right

MyAccountPanel	
Set ManagePanel Set SellBookPanel Set TransactionPanel Set ModificationPanel SetCustomizePanel	CustomizePanel MainFrame ManagePanel ModificationPanel SellBookPanel TransactionPanel User

Figure 3.3.19 MyAccountPanel CRC card

Class responsibilities are listed on the left and collaborators required to accomplish these responsibilities are listed on the right

PicturePanel	
Display image	GUIFactory

Figure 3.3.20 PicturePanel CRC card

Class responsibilities are listed on the left and collaborators required to accomplish these responsibilities are listed on the right

RegisterPanel	
Create user Verify fields Set confirmation panel	ConfirmationPanel GlassPanel GUIFactory MainFrame ServerAccessPoint

Figure 3.3.21 RegisterPanel CRC card

Class responsibilities are listed on the left and collaborators required to accomplish these responsibilities are listed on the right

SearchBooksPanel	
Buy book Display book Search book	GlassPanel GUIFactory MainFrame ServerAccessPoint

Figure 3.3.22 SearchBooksPanel CRC card

Class responsibilities are listed on the left and collaborators required to accomplish these responsibilities are listed on the right

ServerAccessPoint	
Submit SQL statement to database	

Figure 3.3.23 ServerAccessPoint CRC card

Class responsibilities are listed on the left and collaborators required to accomplish these responsibilities are listed on the right

SignedRequestHelper	
Generate Amazon request	

Figure 3.3.24 SignedRequestHelper CRC card

Class responsibilities are listed on the left and collaborators required to accomplish these responsibilities are listed on the right

TransactionPanel	
Display book sales pending Display book purchases pending	GUIFactory ServerAccessPoint

Figure 3.3.25 Transactionpanel CRC card

Class responsibilities are listed on the left and collaborators required to accomplish these responsibilities are listed on the right

User	
Store user information	

Figure 3.3.26 AdminPanel CRC card

Class responsibilities are listed on the left and collaborators required to accomplish these responsibilities are listed on the right

3.5. Refactoring Process

Code Before

LoginPanel.java | MainFrame.java | RegisterPane... | SearchBooks...

```
148
149         panel_13 = new JPanel();
150         panel_12.add(panel_13, BorderLayout.NORTH);
151
152         panel_14 = new JPanel();
153         panel_12.add(panel_14, BorderLayout.CENTER);
154         panel_14.setLayout(new BorderLayout(0, 0));
155
156         panel_15 = new JPanel();
157         panel_14.add(panel_15, BorderLayout.NORTH);
158
159         panel_16 = new JPanel();
160         panel_14.add(panel_16, BorderLayout.CENTER);
161         panel_16.setLayout(new BorderLayout(0, 0));
162
163         panel_17 = new JPanel();
164         panel_16.add(panel_17, BorderLayout.NORTH);
165
166         panel_18 = new JPanel();
167         panel_16.add(panel_18, BorderLayout.CENTER);
168         panel_18.setLayout(new BorderLayout(0, 0));
169
170         panel_19 = new JPanel();
171         panel_18.add(panel_19, BorderLayout.NORTH);
172
173         panel_20 = new JPanel();
174         panel_18.add(panel_20, BorderLayout.CENTER);
175         panel_20.setLayout(new BorderLayout(0, 0));
176
177         panel_21 = new JPanel();
178         panel_20.add(panel_21, BorderLayout.NORTH);
```

Code After

GUIFactory.java | LoginPanel.java | MainFrame.java | RegisterPane...

```
60      * @return
61      * JPanel containing the created FormPanel.
62      */
63     public static JPanel[] createFormPanel(ArrayList<? extends
64         JPanel mainForm = new JPanel(new BorderLayout());
65
66         JPanel panelPtr = mainForm;
67
68         for(int i = 0; i < leftSide.size(); i++){
69             JPanel subForm = new JPanel(new BorderLayout());
70
71             JPanel sectionPanel = new JPanel();
72             sectionPanel.add(leftSide.get(i));
73             sectionPanel.add(rightSide.get(i));
74             panelPtr.add(sectionPanel, BorderLayout.NORTH);
75
76             panelPtr.add(subForm, BorderLayout.CENTER);
77
78             panelPtr = subForm;
79         }
80
81         if(singleComponent != null){
82             JPanel singlePanel = new JPanel();
83             singlePanel.add(singleComponent);
84             panelPtr.add(singlePanel, BorderLayout.NORTH);
85         }
86
87         //return the outermost panel and inner most panel)
88         JPanel[] panels = {mainForm, panelPtr};
89         return panels;
90     }
```

Figure 3.5.1. Refactoring code example

Refactoring is essential for creating code that is maintainable and testable. If code is hard to follow or poorly documented, testing and maintenance can prove almost impossible. In the original code for the application, panels had been put together haphazardly and as a result it was nearly impossible to edit a panel once it had been built. In the newly refactored version of the application, code is far more readable and adaptable. Once the refactoring process had been completed for the applications code, GUIs could be built more efficiently and the end result was a more maintainable and testable application.

4. Technical Documentation

4.1. Programming Languages

The entirety of this software is written in *Java*. *Java* was chosen for a variety of reasons, one being the ability to easily create professional graphical user interfaces with ease through using *Swing*. *Swing* is an object-oriented framework that allows users to design the user layer of the application. Many of the design patterns and abstract classes were designed with *Swing* as a base. This made maintaining and testing our code much easier.

Additionally, *Java* can be run on nearly all platforms. This allows our application to target an extremely large audience without having to worry about technical difficulties specific to a user platform.

4.2. Reused Algorithms and Programs

This section samples the design patterns used throughout the UNH Book Exchange Project. The patterns will be discussed in the following order:

1. Abstractor Design Pattern
2. Factory Design Pattern
3. Observer Design Pattern
4. Singleton Design Pattern

Abstractor Design Pattern

```
/**
 * An abstract class the provides methods for checking text fields. This includes
trimming fields, checking to see if they're valid, checking field
 * length, clearing fields, and setting all input fields to a single button.
 * @author Team11
 *
 */
public abstract class InputPanel extends JPanel {
    /**
     * The labels displayed next to the input fields
     */
    protected ArrayList<JLabel> labels;

    /**
     * The input fields for the panel.
     */
    protected ArrayList<JTextField> textFields;

    /**
     * Initialize array list of input fields.
     */
    public InputPanel() {
        labels = new ArrayList<JLabel>();
        textFields = new ArrayList<JTextField>();
    }

    /**
     * Trims all text fields EXCEPT password fields.
     */
    protected void trimFields(){
        for(int i = 0; i < textFields.size(); i++){
            JTextField jtf = textFields.get(i);
            if(!jtf.getClass().equals(JPasswordField.class)){
                jtf.setText(jtf.getText().trim());
            }
        }
    }
    ...
}
```

Figure 4.2.1. Abstractor Design Pattern

The Abstractor Pattern was chosen because of the number of panels that needed to be used throughout the application. The InputPanel class was created in place of having a separate class for each input panel with code repetition in each class. Additionally, having the InputPanel class abstract allowed for efficient error checking. Since most panels requiring input had to validate that user input, it made sense to simply have every class that needed to validate and accept user input extend an abstract class.

Factory Design Pattern

```
/**
     * Convenience method for creating a FormPanel.
     * @param leftSide
     * All JComponents on the left side of the form.
     * @param rightSide
     * All JComponents on the right side of the form.
     * @param singleComponent
     * Optional single component to be placed at bottom of the form.<br>
     * i.e. Confirmation button
     * @return
     * JPanel containing the created FormPanel.
     */
    public static JPanel[] createFormPanel(ArrayList<? extends JComponent> leftSide,
    ArrayList<? extends JComponent> rightSide, JComponent singleComponent){
        JPanel mainForm = new JPanel(new BorderLayout());

        JPanel panelPtr = mainForm;

        for(int i = 0; i < leftSide.size(); i++){
            JPanel subForm = new JPanel(new BorderLayout());

            JPanel sectionPanel = new JPanel();
            sectionPanel.add(leftSide.get(i));
            sectionPanel.add(rightSide.get(i));
            panelPtr.add(sectionPanel, BorderLayout.NORTH);
            panelPtr.add(subForm, BorderLayout.CENTER);

            panelPtr = subForm;
        }

        if(singleComponent != null){
            JPanel singlePanel = new JPanel();
            singlePanel.add(singleComponent);
            panelPtr.add(singlePanel, BorderLayout.NORTH);
        }

        //return the outermost panel and inner most panel)
        JPanel[] panels = {mainForm, panelPtr};
        return panels;
    }
```

Figure 4.2.2. Factory Design Pattern

The factory design pattern is designed to accept input and churn out varying output, much like a factory in the real world. The above method, createFormPanel, does not need to know what the Form Panel looks like, who needs to access it, or any other information about the Form Panel besides what is on it. If a Form Panel was being created without this design pattern, the layout and type of Form would have had to be decided before creation. There are many other similar methods in the GUIFactory class for creating different aspects of the GUI.

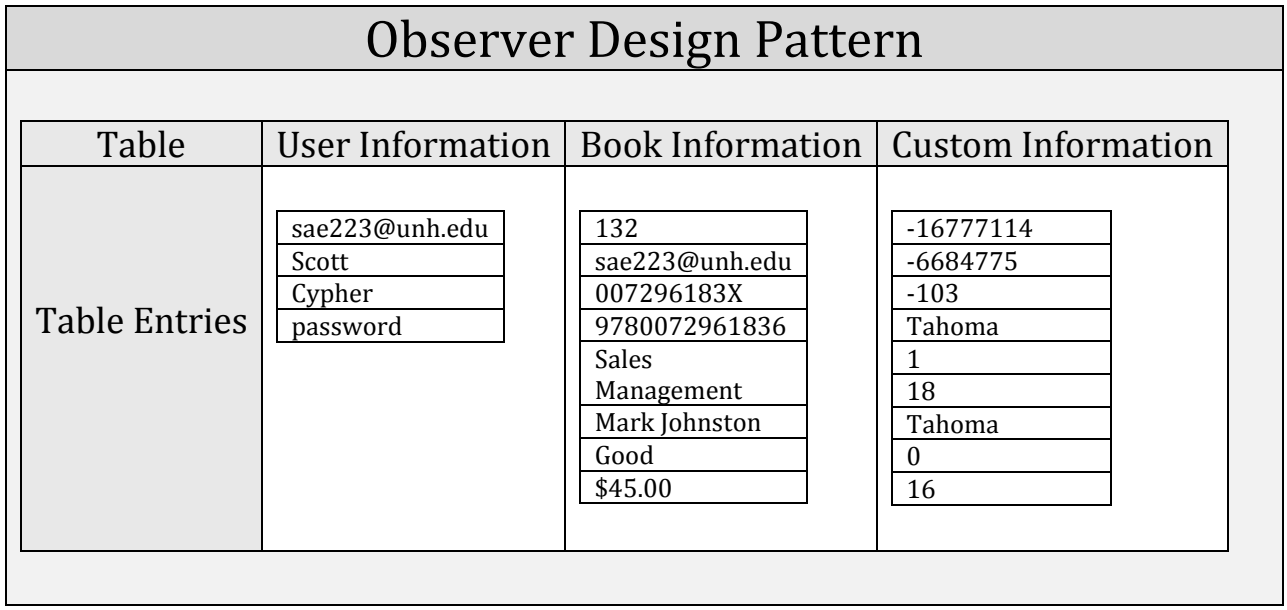


Figure 4.2.3 Observer Design Pattern

The observer pattern consists of an object notifying all of its "observers" whenever there is a state change. While this isn't implemented directly in code, our SQL database is a perfect example of an implementation of this pattern. We related the primary key of "UserInformation" to the foreign keys of "BookInformation" and "CustomInformation". Once the one-to-many relationships were established, the observer pattern was then put in place. If a user was deleted from the database by an administrator, all of the records involving that user in the other two "observers", in this case "BookInformation" and "CustomInformation" would be updated accordingly. An example can be seen by viewing the above tables. When sae223@wildcats.unh.edu is deleted from the User Information database, all other records in the database would be deleted because of the Observer design pattern within the relational database design.

Singleton Design Pattern

```
/**
 * The communication medium between the application and the SQL server.
 * The ServerAccessPoint is used to update all tables within
 * the database. The ServerAccessPoint implements the Singleton design pattern
 * in order to ensure that there is only one object capable
 * of modifying the database. The ServerAccessPoint is capable of many SQL operations
 * such as modifying a table entry, deleting a table entry, and adding a table entry.
 * @author Team11
 *
 */
public class ServerAccessPoint {

    /**
     * The single instance of ServerAccessPoint.
     */
    private static ServerAccessPoint sap;

    /**
     * Private constructor for singleton design pattern.
     */
    private ServerAccessPoint(){}

    /**
     *  Returns the ServerAccessPoint.
     */
    public static ServerAccessPoint getInstance(){
        if(sap == null){
            sap = new ServerAccessPoint();
        }

        return sap;
    }
    ...
}
```

Figure 4.2.4 Singleton Design Pattern

The singleton design class was used in order to prevent multiple instantiations of important data accessing objects. ServerAccessPoint is the main class that communicates with our SWL server, and we want to ensure that the class is only instantiated once and then passed around. This provides more assurance with the data validity of the program. The singleton design pattern is preferred over creating a static class because a singleton allows access to a single created instance (or rather, a reference to that instance). That instance can be passed as a parameter to other methods, and treated as a normal object. On the contrary, a static class only allows static methods. The singleton design pattern can also be seen in the MainFrame class and EmailClient class.

4.3. Tools and Environments

The developing environment chosen for this application was *Eclipse*. *Eclipse* is a powerful cross platform integrated development environment (IDE) which provides many tools to make a programmer much more efficient. One of the major benefits of *Eclipse* is its ability to compile code while the code is being written. If the programmer forgets to import something, attempts to access an inaccessible variable, or makes an incorrect method call, *Eclipse* immediately underlines the error in red. By simply scrolling over the area in red, *Eclipse* will provide a list of possible solutions that could fix the compilation error. In the situation where an import was forgotten, *Eclipse* would automatically import the needed package from the java framework if this fix was selected from the generated list.

An additional reason *Eclipse* was chosen are the many downloadable plug-ins. One plug-in that was especially helpful was Subclipse. This allowed us to connect and update SVN with our project with ease, allowing for group members to always stay on the same page with a program. Another plug-in that was helpful was UMLLet. UMLLet is further documented in Section 7, UML Tool documentation.

Lastly, *Eclipse* was chosen because of the ability to use JUnit testing. JUnit testing is a Java testing framework that allows an application to be repeatedly tested using automated tests. This proved extremely beneficial to the applications success because whenever large updates were made to the program, a previously established series of JUnit tests could be performed to ensure that no previous functionality of the program had been lost.

4.4. Database Tables

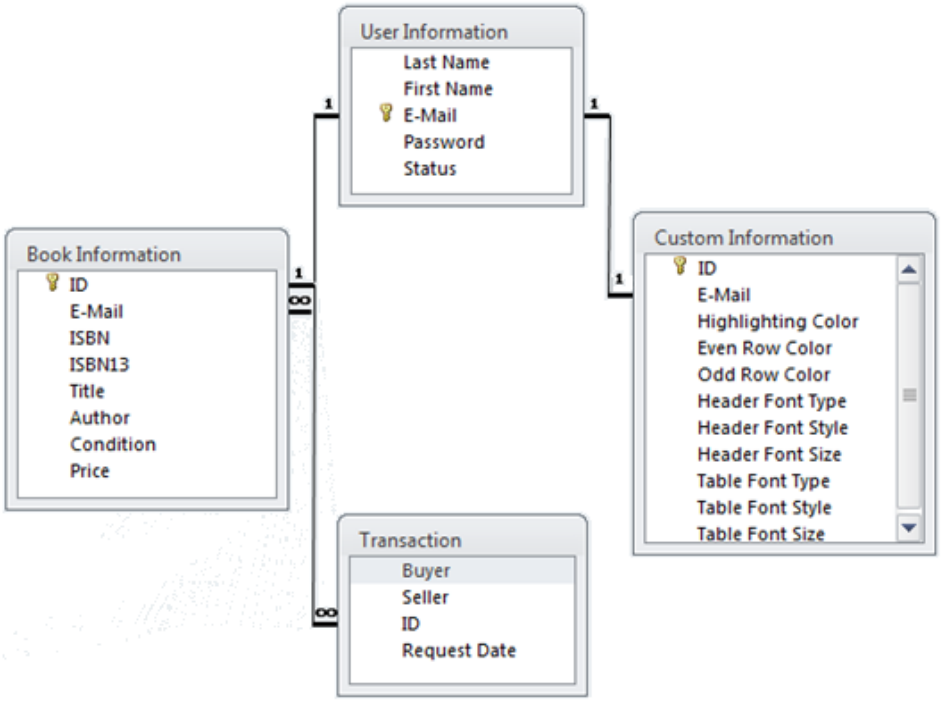


Figure 4.4.1: Relational Database Design

The above database tables are accessed through *mySQL*. The relationships depicted above are of one-to-many and one-to-one. Each user can have multiple books, but each book is associated with a single user. Each book can have multiple transactions, but each transaction is associated with a single book. Lastly, each user can have one custom information and each custom information is associated with a single user. These relationships are established in SQL through the declaration and connection of primary and foreign keys, as depicted above. By establishing this relationship, the database ensures that when deleting is cascaded throughout the relational database tree.

Book Information		
NAME		**TYPE**
ID	**Primary Key**	int(11)
Email	**Foreign Key**	varchar(40)
ISBN		varchar(40)
ISBN13		varchar(40)
Title		Varchar(40)
Author		varchar(40)
Condition		varchar(40)
Price		varchar(40)

Figure 4.4.1Book Information Table

The Book Information table stores all necessary information needed to display a book in the application.

User Information		
NAME		**TYPE**
E-Mail	**Primary Key**	varchar(40)
Password		varchar(40)
Last Name		varchar(40)
First Name		varchar(40)
Status		int(1)

Figure 4.4.2User Information Table

The User Information table stores all account information associated with a user.

Custom Information		
NAME		**TYPE**
ID	**Primary Key**	varchar(40)
E-Mail	**Foreign Key**	varchar(40)
Even Row Color		varchar(40)
Odd Row Color		varchar(40)
Header Font Type		varchar(40)
Header Font Style		varchar(40)
Header Font Size		varchar(40)
Table Font Type		varchar(40)
Table Font Style		varchar(40)
Table Font Size		varchar(40)

Figure 4.4.3 Custom Information Table

The CustomInformation table stores all customization information the user chooses to have. Upon account creation, this table is set to a series of default values that can be changed in the customization panel.

Transaction		
NAME		**TYPE**
ID	**Primary Key**	varchar(40)
E-Mail	**Foreign Key**	varchar(40)
Even Row Color		varchar(40)
Odd Row Color		varchar(40)

Figure 4.4.4 Transaction Table

The Transaction table stores all records of books that a user has requested to buy and requests for a book that a user is selling. If a book is deleted, the offer will be not shown in the transaction history. This allows the user to keep track of books being bought and sold even though it is done through email.

5. User Documentation

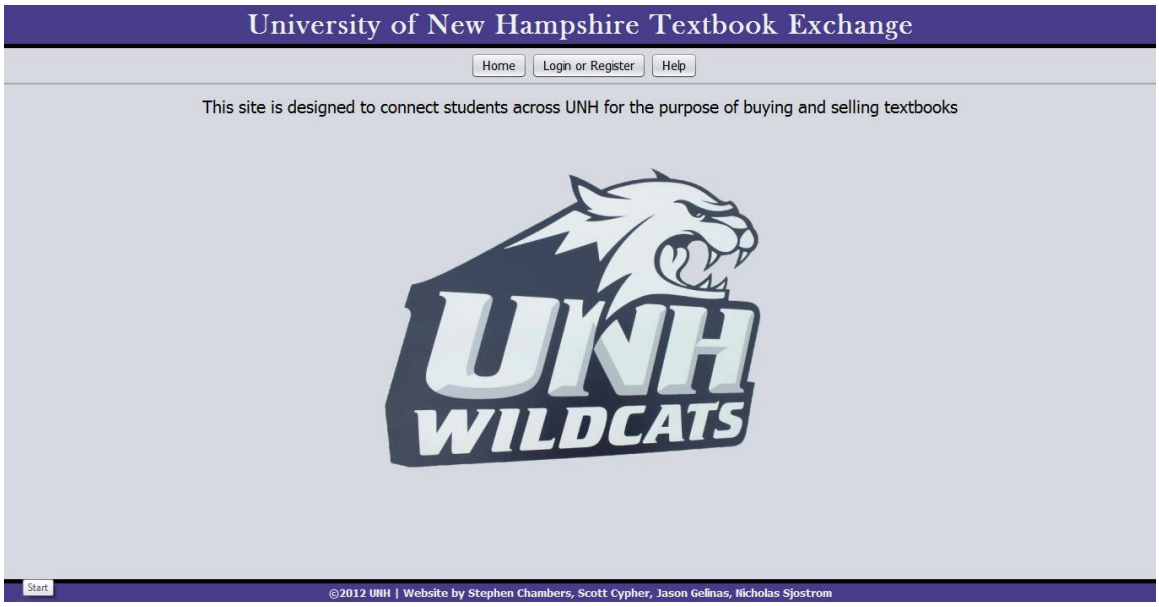


Figure 5.1: Home Screen

When the application is opened the home screen is displayed. The home screen contains a UNH Wildcats photo and a message to the user containing the site description. All of the buttons except "Home", "Login or Register", and "Help" are hidden at this point of the application. The user then clicks "Login or Register" or "Help" to continue.

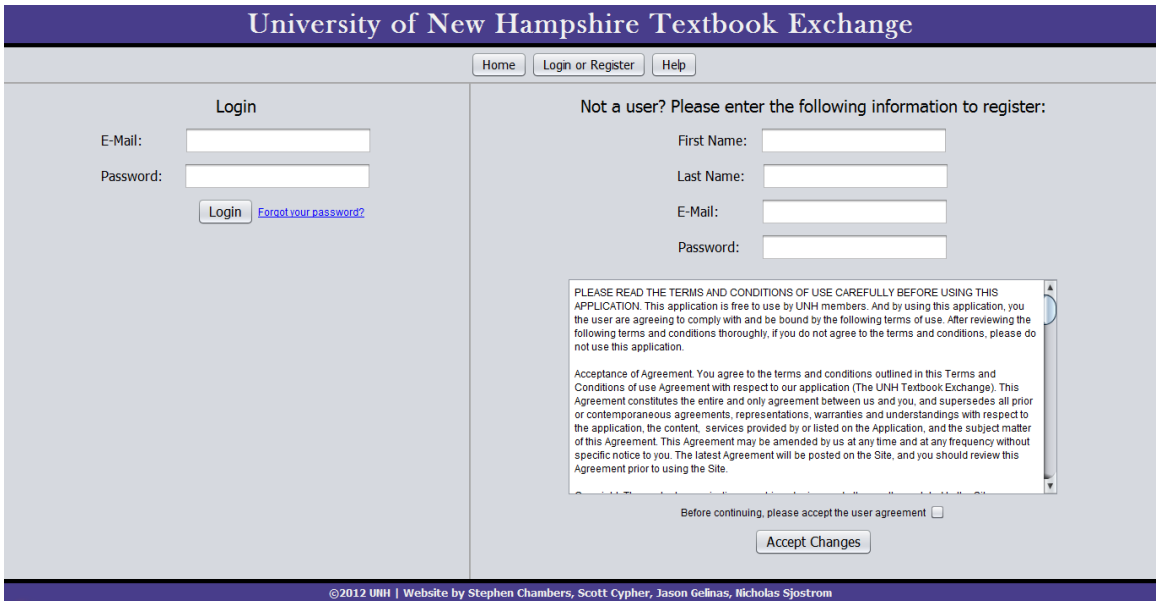


Figure 5.2: Login/Registration Screen

Login: If the user already has an account, they can log in on the left side of the screen. All that is required is their UNH email and password. The server then verifies that the user exists in the database, and if they do,

they are brought to the My Account screen. If they are not verified, the system displays a message conveying this to the user.

Register: The user provides the system with their First Name, Last Name, Email and Password. The user then reads the terms of agreement, and if they are agreeable, clicks the "Accept the user agreement" checkbox. The user then clicks on "Finish Registration", which brings them to the confirmation screen.

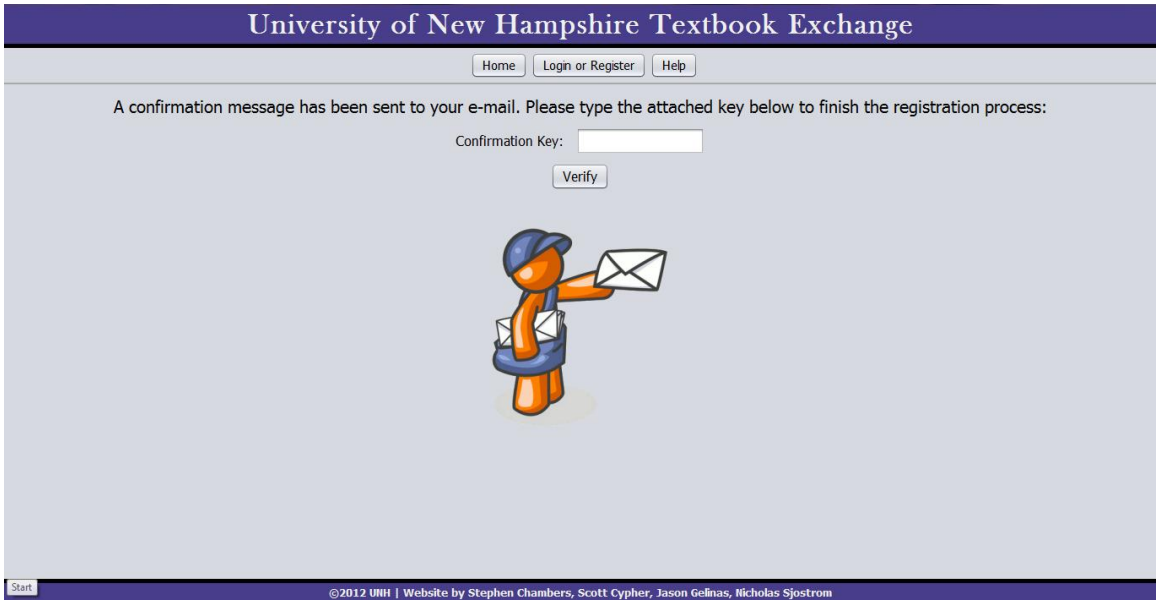


Figure 5.3: Confirmation Screen

The confirmation screen is displayed when the user registers in the system for the first time *or* when the user changes their email through the Modification Screen. When this screen is displayed, a thread is generated whose main responsibility is to generate a random verification key and send an email to the address provided by the user in the previous step. The thread improves the user experience by not slowing down the flow of the application while sending the email. If the verification key entered is not correct, the system displays this to the user. If it is correct, the confirmation screen is displayed letting the user know they have successfully either registered or changed their email address in the system.

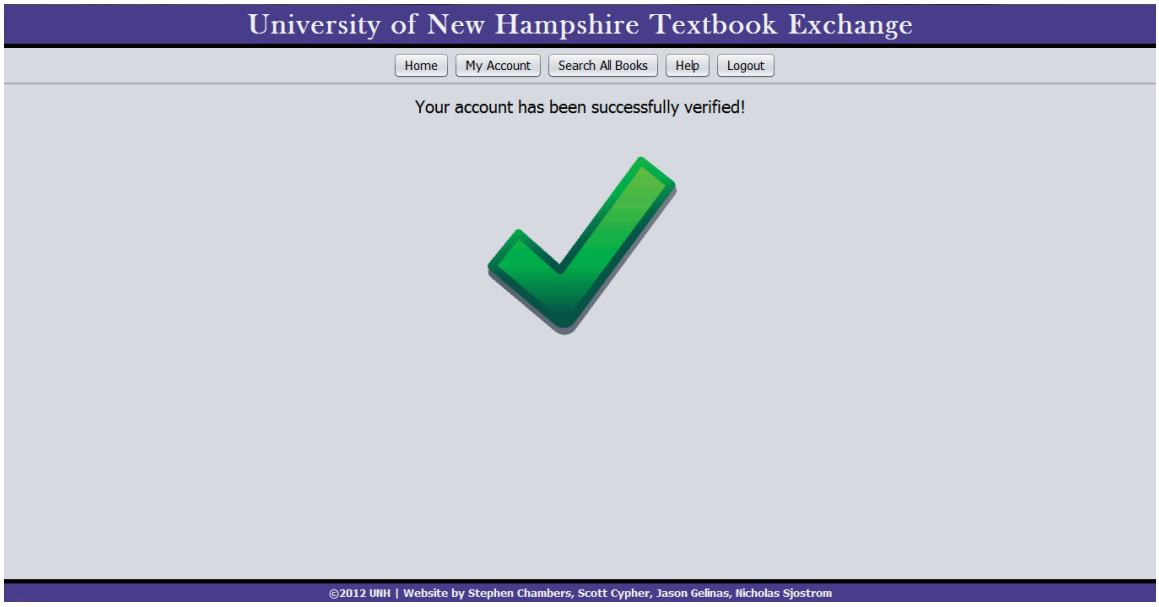


Figure 5.4: Success Screen

The confirmation screen is displayed to notify the user their task has been successfully completed by the application. The text at the top of the screen can be adapted based on the actions that the user completed. The user can then click on the "My Account" button to be brought to the My Account screen to get started.

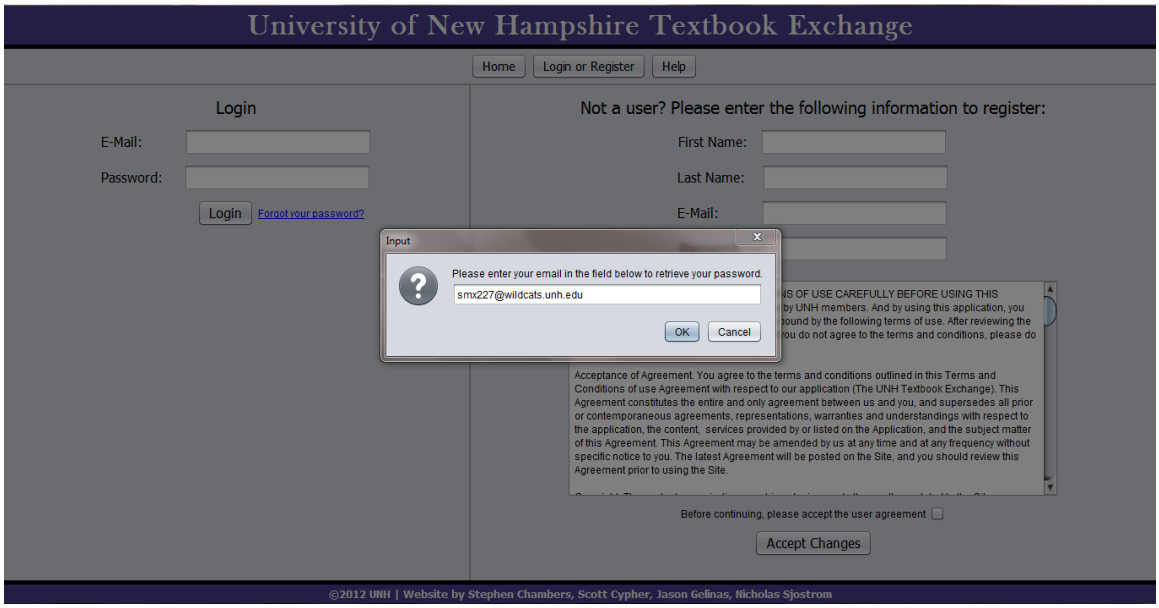


Figure 5.5: Forgot Password Feature

If the user forgets their password, they can click on the "Forgot your password?" link. Notice how when the system communicates with the user, the surrounding application is dimmed and becomes un-clickable. The cursor also adapts when the link is scrolled over the "Forgot your password?" link to give the application a classic feel. In the system prompt, the user can enter an email address. If the email address exists in the database, the server will email the user the unique password linked with that email. If it does not, the system will display a message letting the user know that the email was not found in the database.

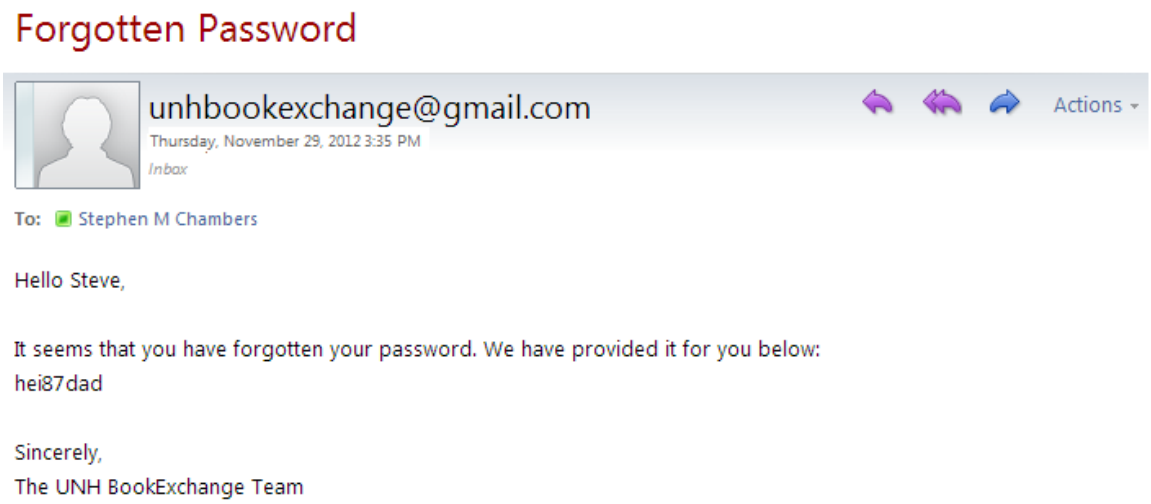


Figure 5.5: Forgot Password Email

This is the email sent by the system when the user enters a valid email address in the system prompt after clicking on the "Forgot your password?" link.

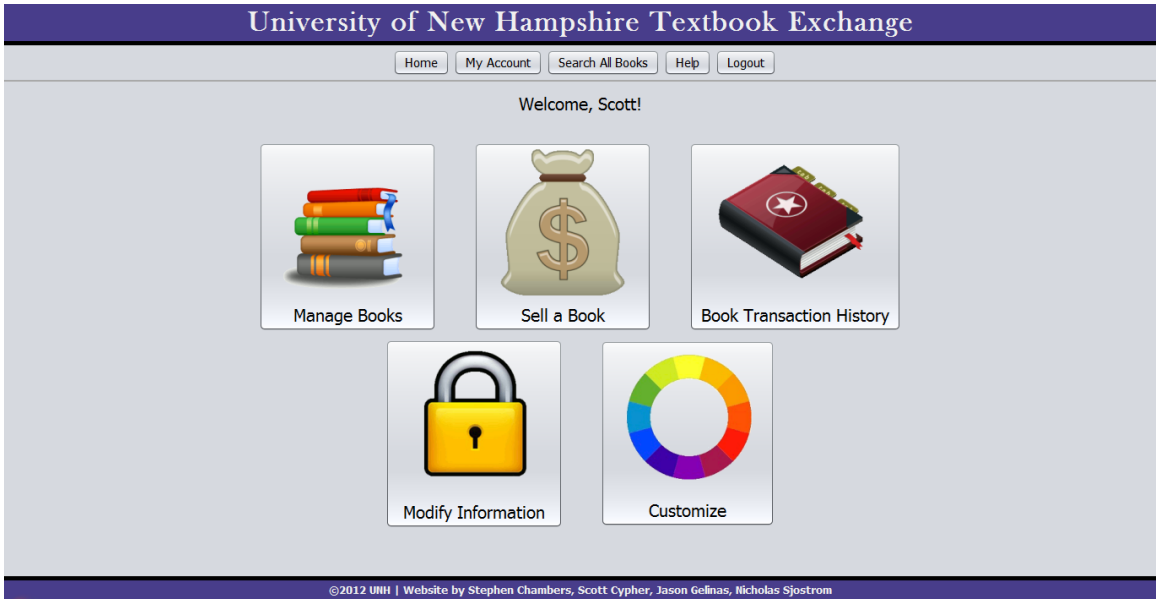


Figure 5.6: Manage Account Screen

This screen is the crux of the user experience. The user accesses the majority of the sites functionality through this screen by clicking on the five different image icons displayed above. Image icons were chosen to enrich the user experience. For the rest of the user documentation section, this guide will assume that user clicks on the "My Account" button to navigate to other features of the site.

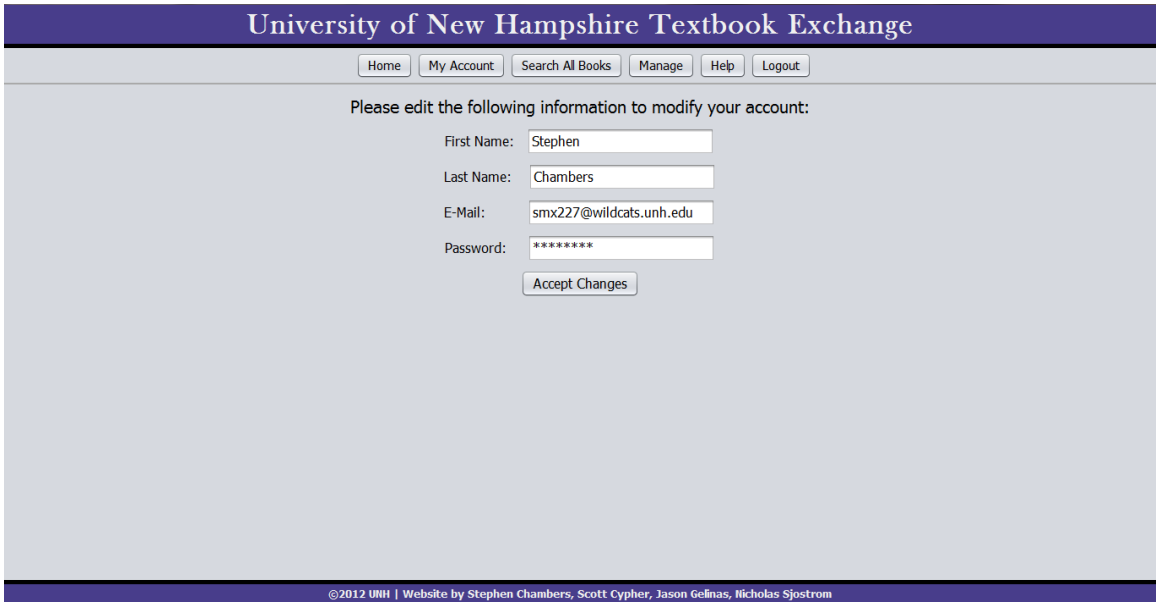


Figure 5.7: Modification Screen

If the user's registered information changes at any point during the user's time with the application, modifying that change with the site is easy. When the screen is loaded, the server fetches all current user information and displays them corresponding easy to modify text fields. If the user wants to change their information, they simply click inside the text field, change the information, and click accept changes. The server processes the user request and updates the data accordingly.

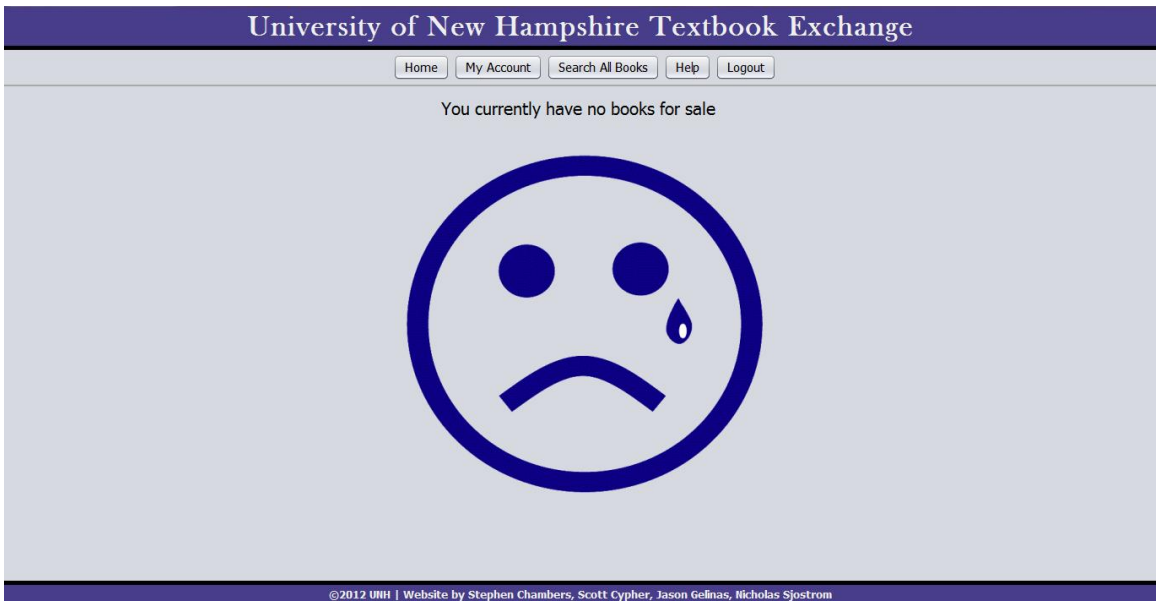


Figure 5.8: Mange Books Screen Pre-Sale

Before selling a book, the manage book screen will inform the user that there are no books for sale. If a book is sold, this screen will be populated with the books currently sold by the user.

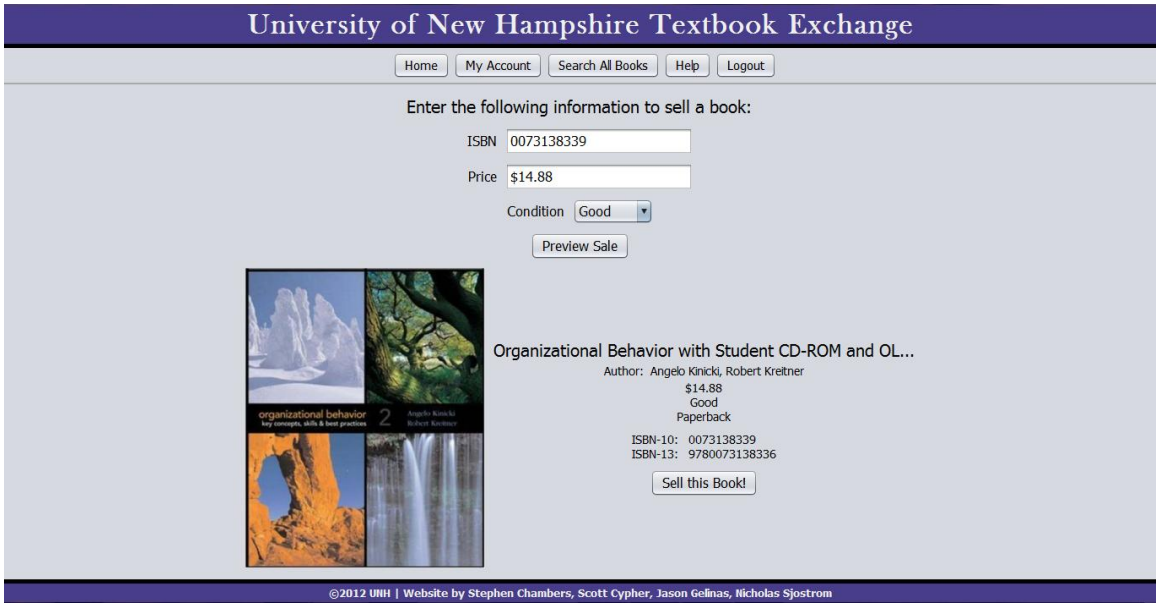


Figure 5.9: Sell Book Screen

If the user wishes to post a book onto the system, all they need to know is either the ISBN or the ISBN13 of the book that they want to post. Once it is entered, they must enter the price at which they want to sell the book and the current condition of the book. When preview sale is hit, the sale as it would look to an outside user once the book is in the system is displayed below. If the user is satisfied with the sale, they click the "Sell this Book!" button to post the book onto the system.

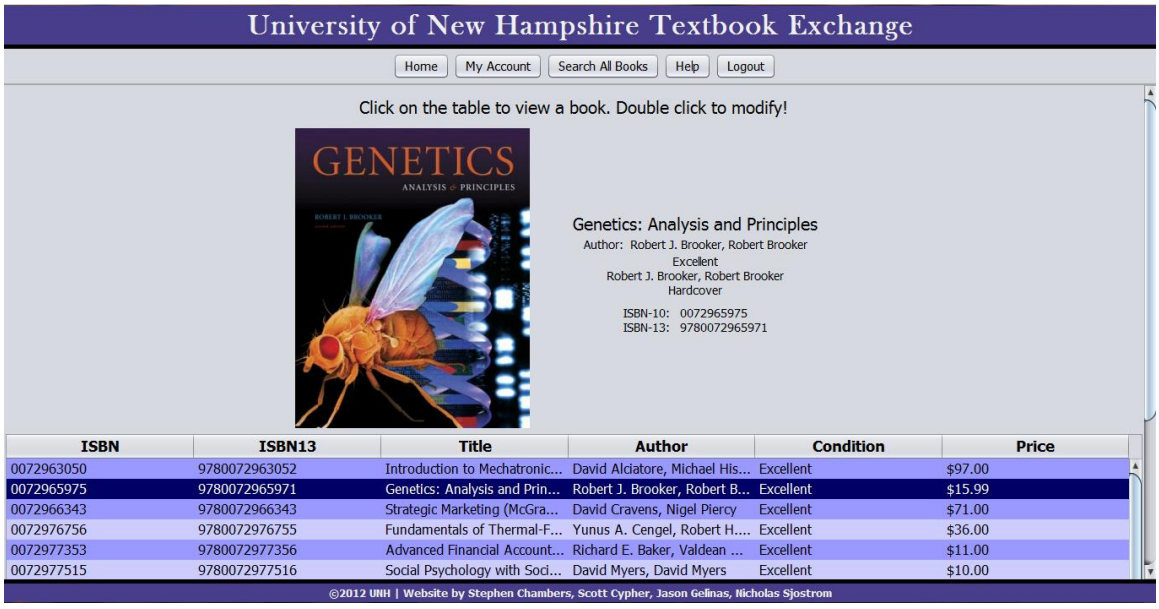


Figure 5.10: Mange Books Screen Post-Sale

Once the user sells a book, they can go to the Manage Books screen in order to modify their book information or delete the book from the database. If the user clicks on an item in the table, the corresponding book image and information appear above the table.

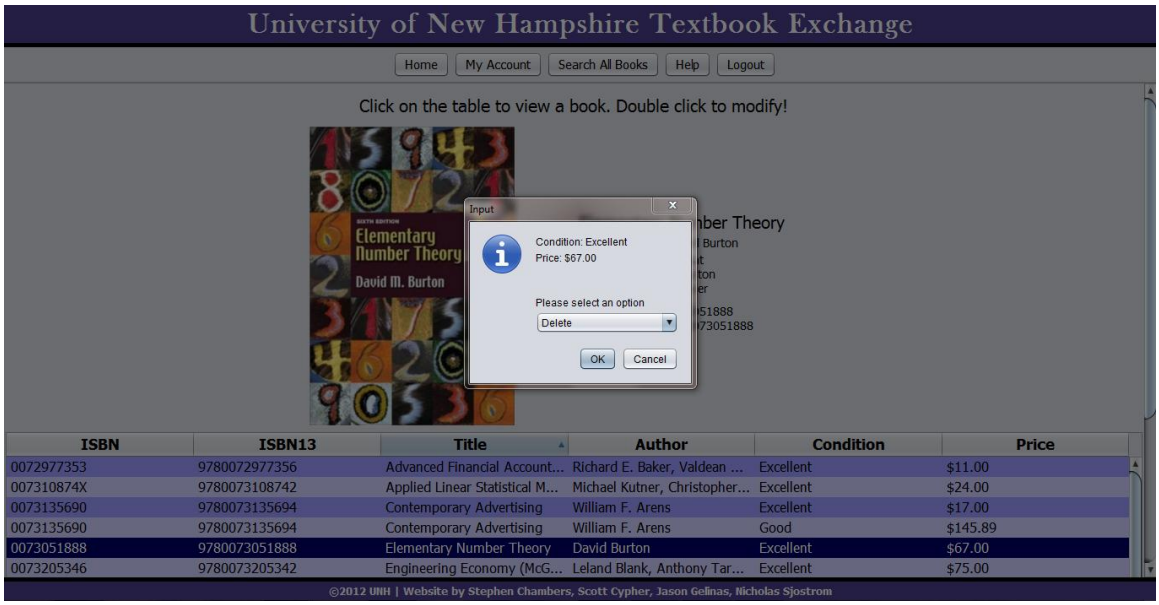


Figure 5.11: "Double click to modify" Feature

If the user wants to delete or modify their book, they simply on the double click on the book in the table shown on the screen. They can then choose "Delete" or "Modify" from a drop down menu and follow the steps prompted by the server to complete the process.

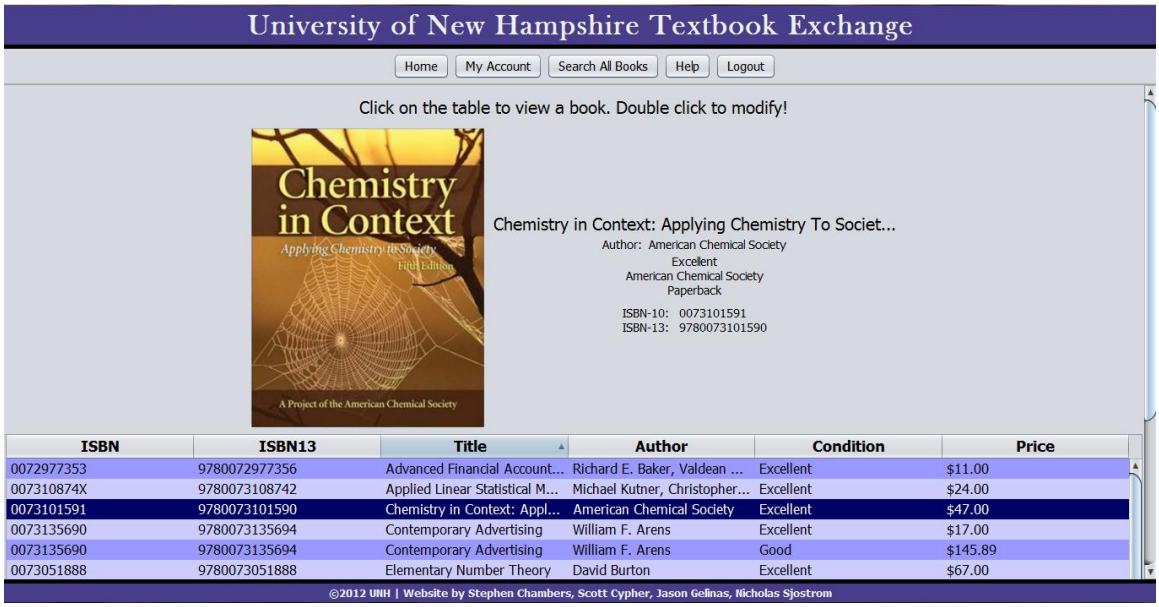


Figure 5.12: Table sort feature

If the user wants to sort based on a certain criteria, it can be done by simply double clicking on the header of the field you want to sort by. The table will default to sorting by descending order, however, the table can be sorted by ascending order if clicked again. Regardless of how the table is sorted, the correct book will display above the table based on the row that the user selected.

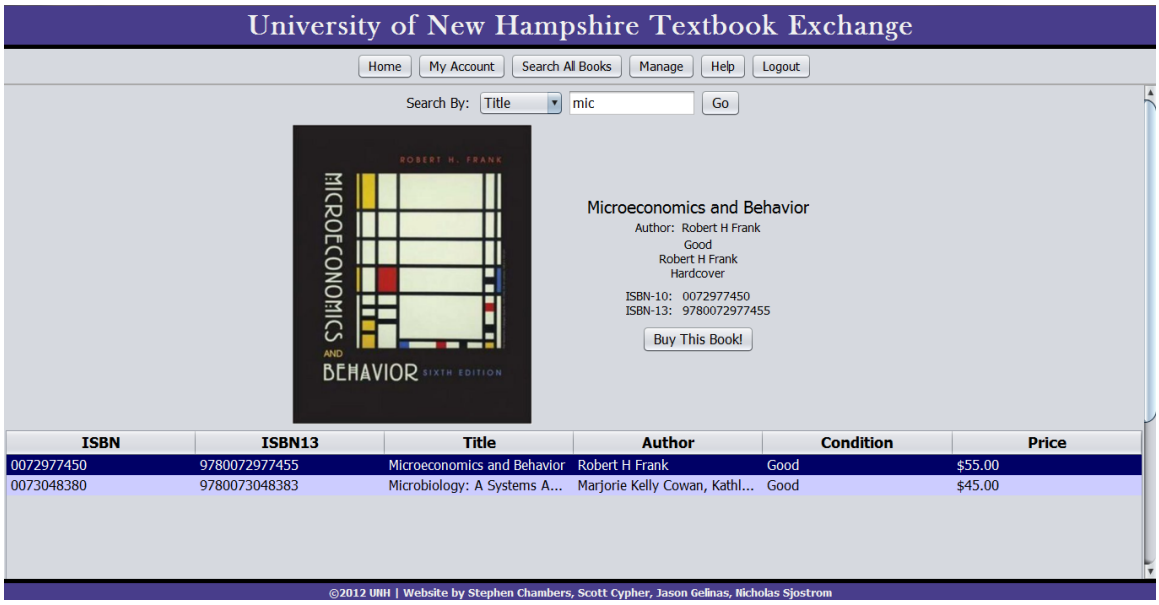


Figure 5.13: Search Book Screen

The user can also browse the books stored in the database. A drop down menu specifying the category to search is displayed at the top of the screen. The user selects the category, and specifies their search within the textbox to the right of the drop down menu. The search does not need to exactly match the entry in the server, as demonstrated above. Microbiology and Microeconomics books were displayed when "mic" was searched within the title category. After specifying their search, the user clicks the "Go" button, and the query is sent to the database. The database returns all books matching the search parameters and displays them in a table similar to the one that displayed the users sold books. The user can click on any item in the table to display book information.

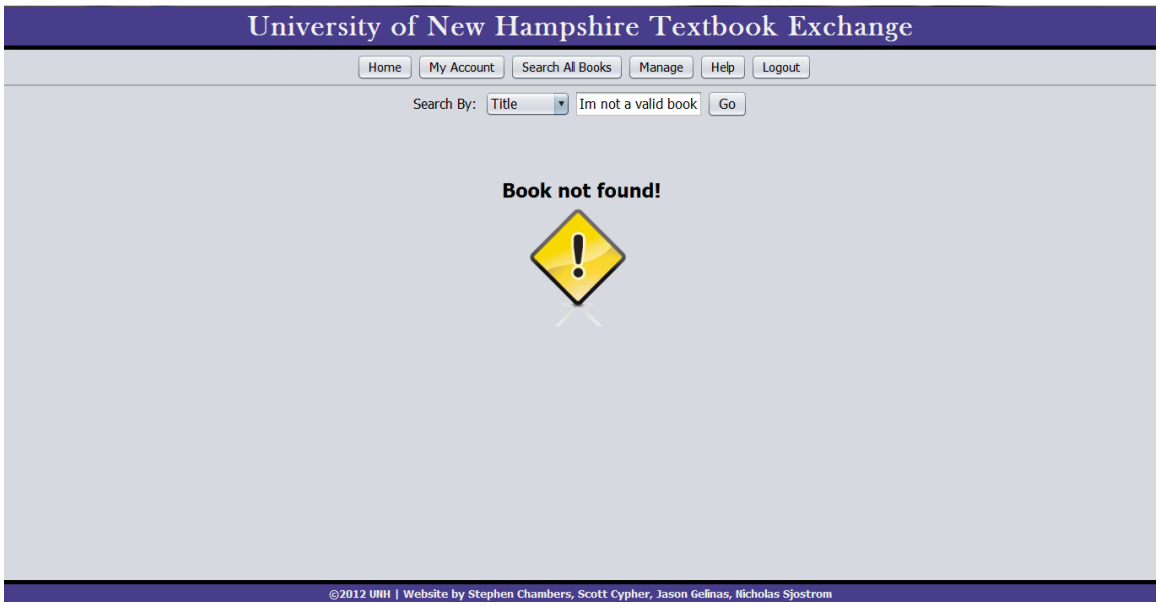


Figure 5.13: Book not found

If the book cannot be found in the database, a message is displayed to the user indicating this.

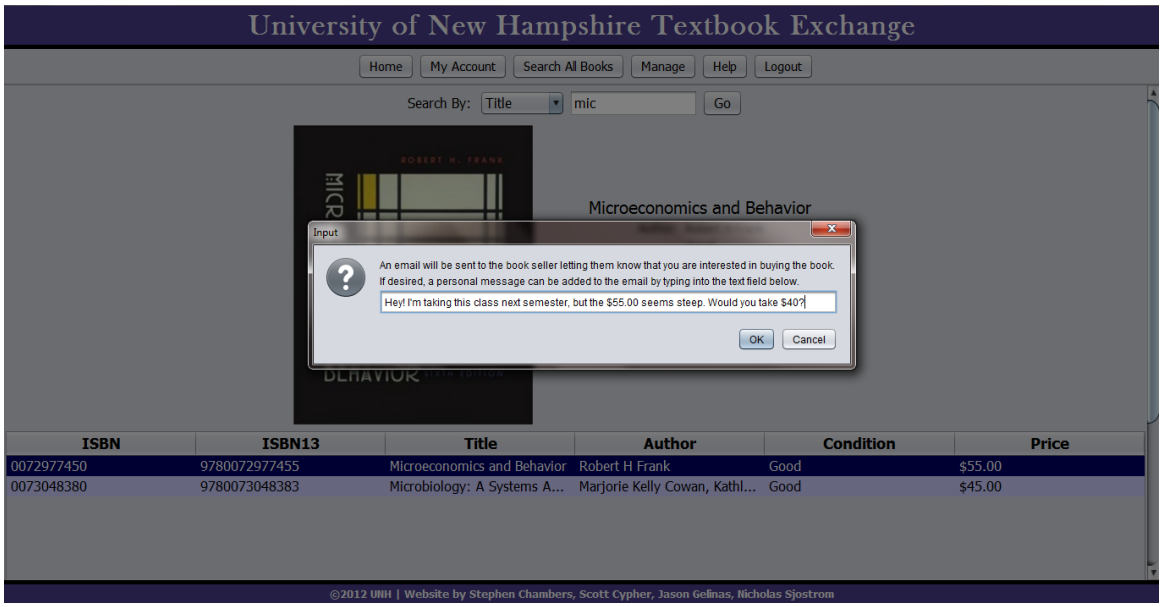


Figure 5.14: Actually buying a book

If the user is interested in buying the book that was searched for, they click the "Buy this Book!" button. The system then informs the user that an email will be sent to the seller of the book. The user can also enter a personal message along with their email. Once O.K. is clicked, a thread spawns and sends the email out to the seller. The user cannot buy their own book or contact the seller of a particular book more than once.

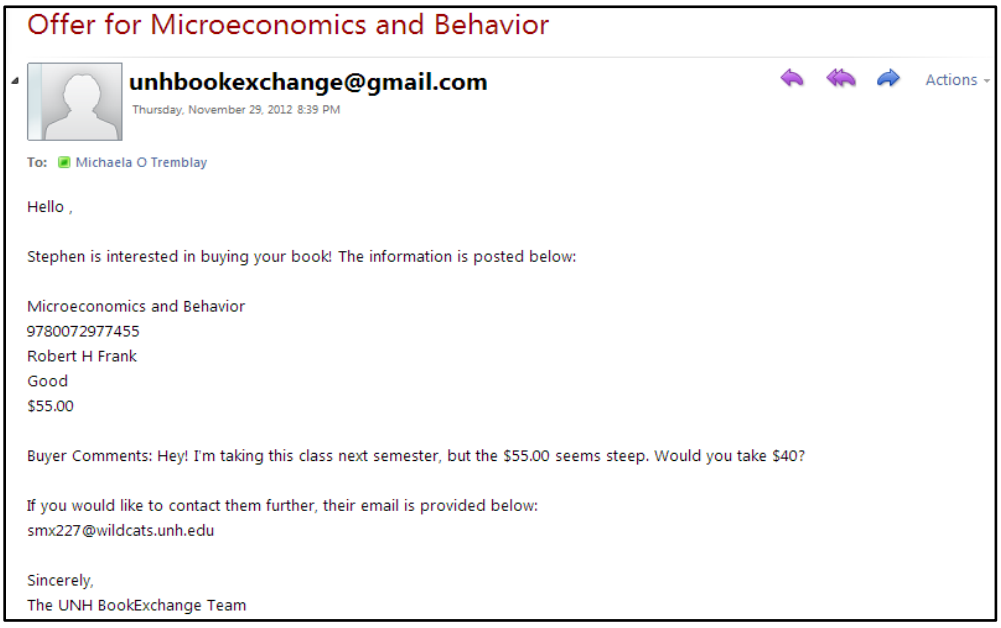


Figure 5.15: Actually buying a book: Email sent

This email informs the seller of the book that someone has interest in buying it. The book information is displayed along with the personal message and further contact information.

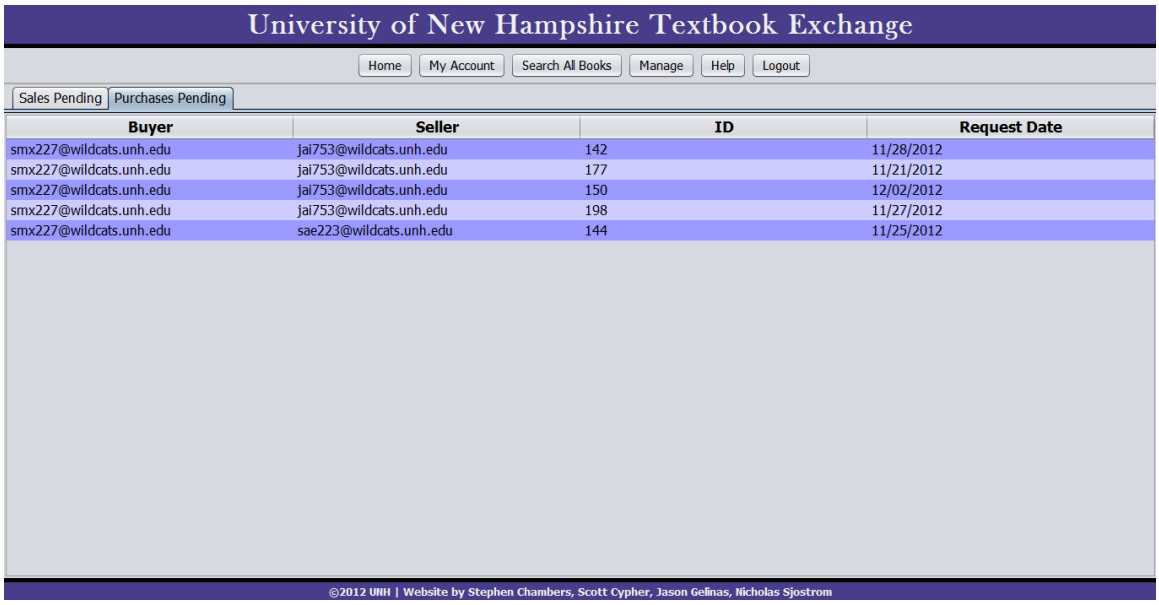


Figure 5.16: Transaction History: Purchases Pending

Once the email is sent to the seller, it will show up in the "Purchases Pending" portion of the transaction history. The user can see who the seller of the book is, the book identification number that is requested, and the date that the email was sent.

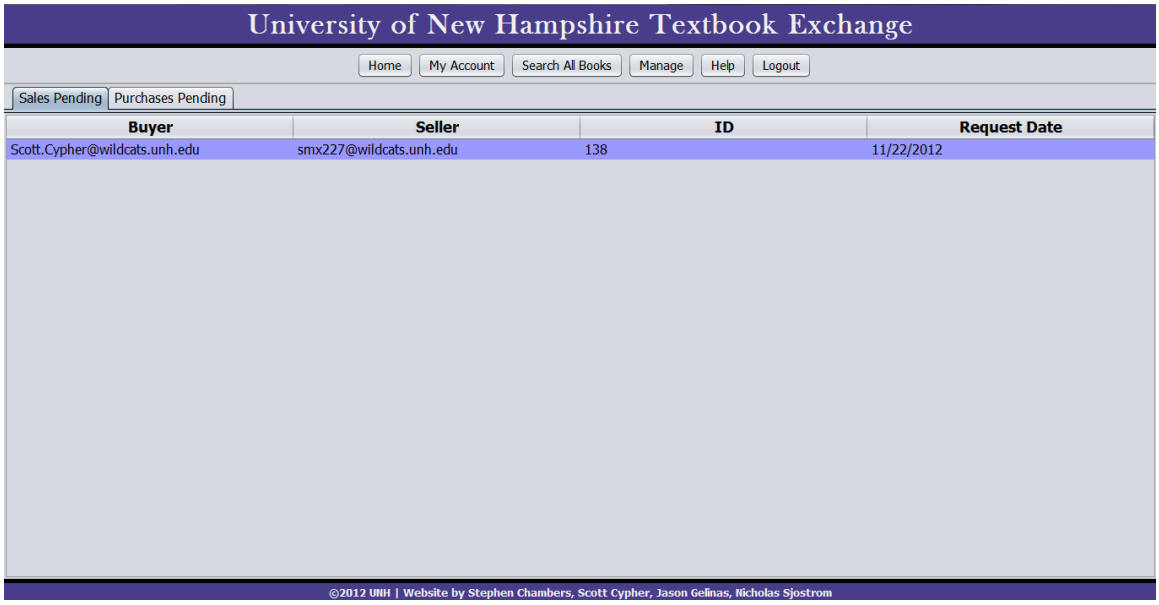


Figure 5.17: Transaction History: Sales Pending

Once an email is sent to the user requesting an external user's interest in a book, it will show up in the "Sales Pending" portion of the transaction history. The user can see who the potential buyer of the book is, the book identification number that was requested, and the date that the email was sent.

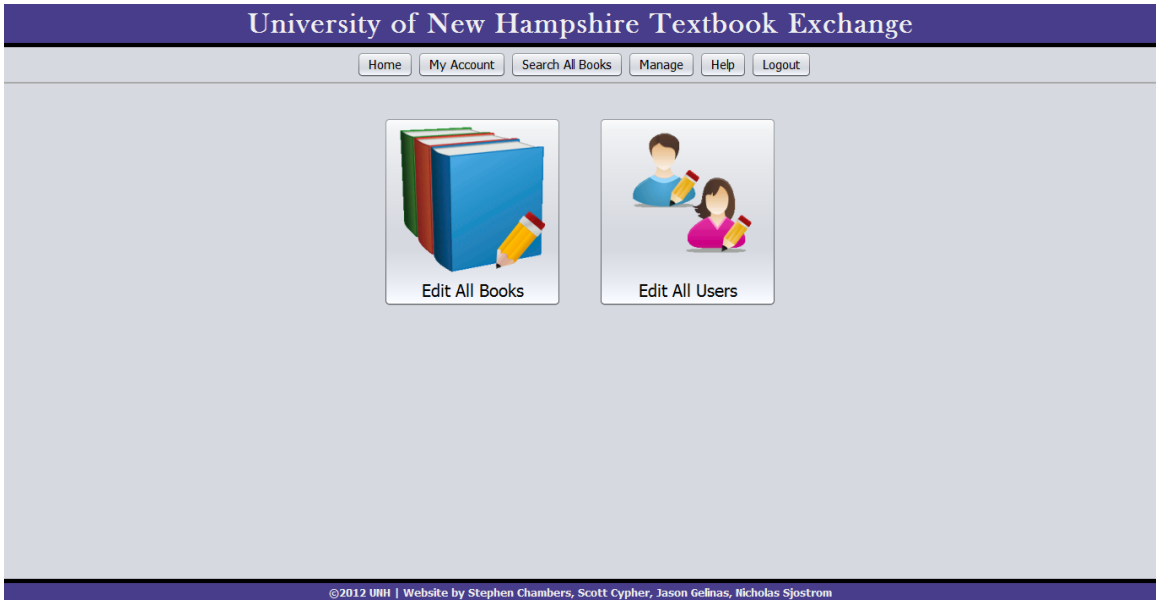
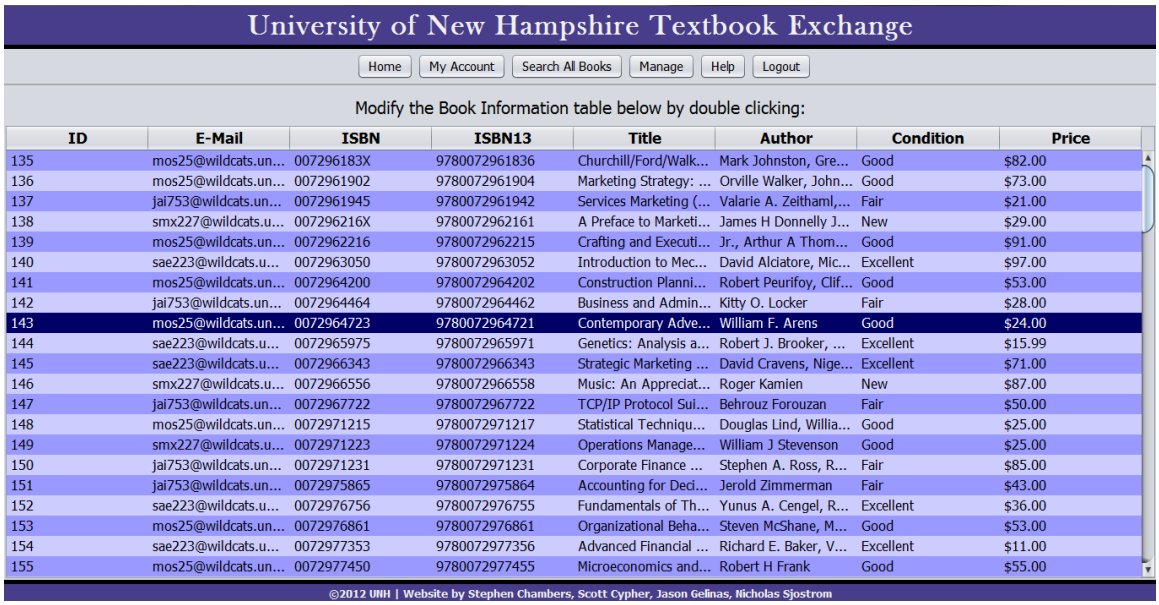


Figure 5.18: Administrative User

If the user is an administrator, they have certain privileges that normal users do not have. The "Manage" button appears when the user logs in and the database confirms their status. When the "Manage" button is clicked, the user can then click the "Display all Users" button or the "Display all Books" button. This brings up a table of all the current users or books in the database.



ID	E-Mail	ISBN	ISBN13	Title	Author	Condition	Price
135	mos25@wildcats.un...	007296183X	9780072961836	Churchill/Ford/Walk...	Mark Johnston, Gre...	Good	$82.00
136	mos25@wildcats.un...	0072961902	9780072961904	Marketing Strategy: ...	Orville Walker, John...	Good	$73.00
137	jai753@wildcats.un...	0072961945	9780072961942	Services Marketing (...	Valarie A. Zeithaml,...	Fair	$21.00
138	smx227@wildcats.u...	007296216X	9780072962161	A Preface to Marketi...	James H Donnelly J...	New	$29.00
139	mos25@wildcats.un...	0072962216	9780072962215	Crafting and Executi...	Jr., Arthur A Thom...	Good	$91.00
140	sae223@wildcats.u...	0072963050	9780072963052	Introduction to Mec...	David Alciatore, Mic...	Excellent	$97.00
141	mos25@wildcats.un...	0072964200	9780072964202	Construction Planni...	Robert Peurifoy, Clif...	Good	$53.00
142	jai753@wildcats.un...	0072964464	9780072964462	Business and Admin...	Kitty O. Locker	Fair	$28.00
143	mos25@wildcats.un...	0072964723	9780072964721	Contemporary Adve...	William F. Arens	Good	$24.00
144	sae223@wildcats.u...	0072965975	9780072965971	Genetics: Analysis a...	Robert J. Brooker, ...	Excellent	$15.99
145	sae223@wildcats.u...	0072966343	9780072966343	Strategic Marketing ...	David Cravens, Nige...	Excellent	$71.00
146	smx227@wildcats.u...	0072966556	9780072966558	Music: An Appreciat...	Roger Kamien	New	$87.00
147	jai753@wildcats.un...	0072967722	9780072967722	TCP/IP Protocol Sui...	Behrouz Forouzan	Fair	$50.00
148	mos25@wildcats.un...	0072971215	9780072971217	Statistical Techniqu...	Douglas Lind, Willia...	Good	$25.00
149	smx227@wildcats.u...	0072971223	9780072971224	Operations Manage...	William J Stevenson	Good	$25.00
150	jai753@wildcats.un...	0072971231	9780072971231	Corporate Finance ...	Stephen A. Ross, R...	Fair	$85.00
151	jai753@wildcats.un...	0072975865	9780072975864	Accounting for Deci...	Jerold Zimmerman	Fair	$43.00
152	sae223@wildcats.u...	0072976756	9780072976755	Fundamentals of Th...	Yunus A. Cengel, R...	Excellent	$36.00
153	mos25@wildcats.un...	0072976861	9780072976861	Organizational Beha...	Steven McShane, M...	Good	$53.00
154	sae223@wildcats.u...	0072977353	9780072977356	Advanced Financial ...	Richard E. Baker, V...	Excellent	$11.00
155	mos25@wildcats.un...	0072977450	9780072977455	Microeconomics and...	Robert H Frank	Good	$55.00

Figure 5.19: Administrative User: Delete Books

The administrator can delete books by double-clicking on a record and typing "DELETE". The server will then delete the record in the database and it will not appear on the table.

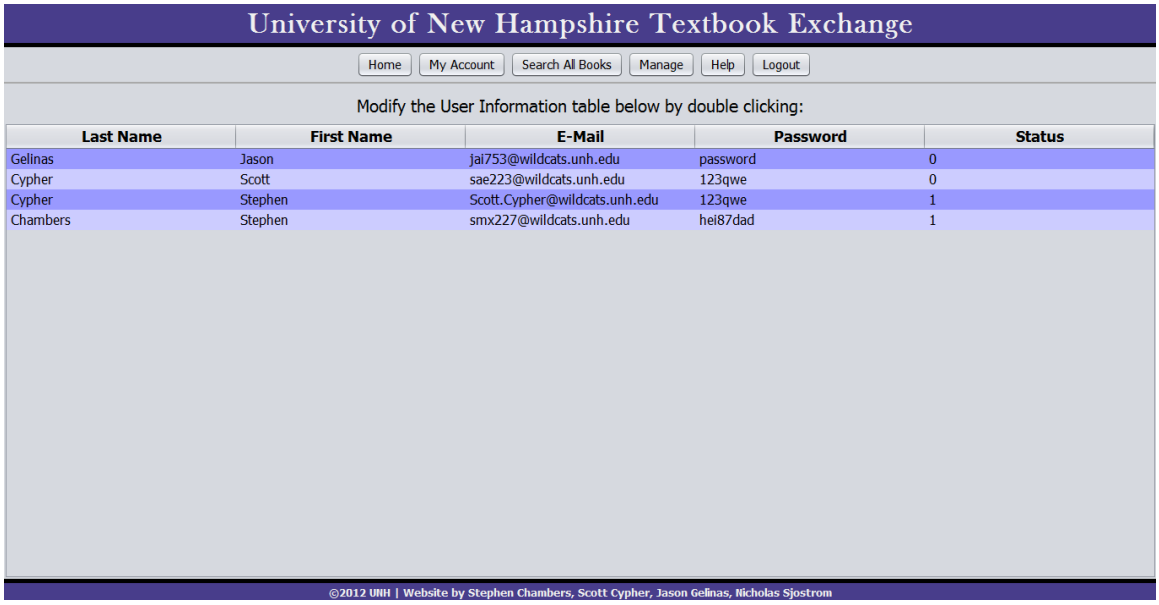


Figure 5.20: Administrative User: Delete/Modify Users

The administrator can delete and modify users using the double clicking feature. An administrator can promote a user to administrative status by changing the value of the status field to 1. However, the administrator cannot delete or modify other administrators.

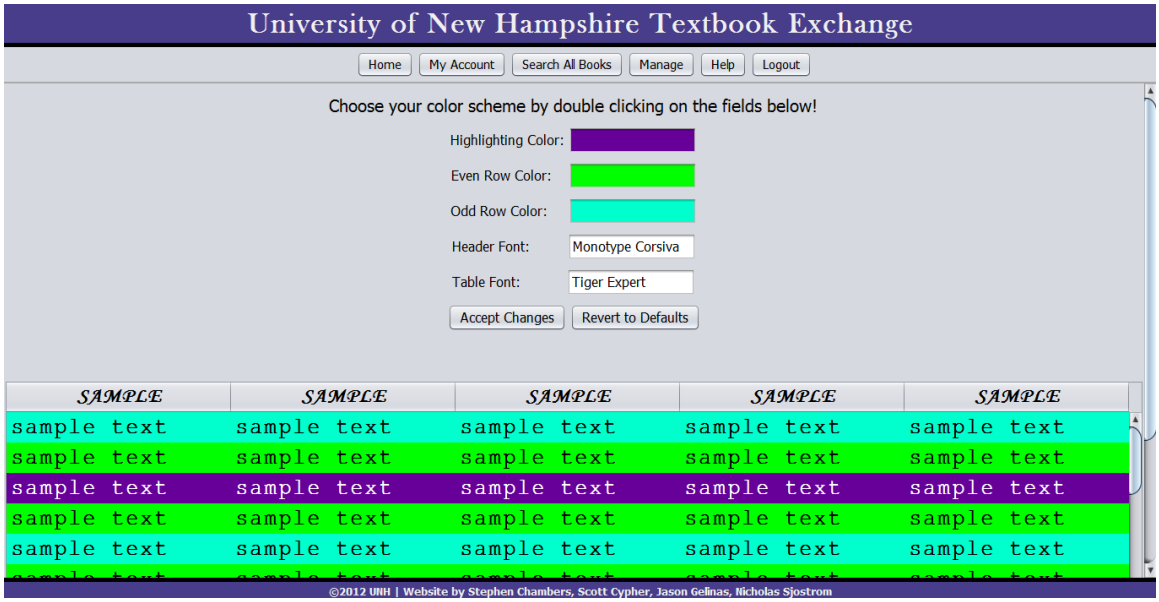


Figure 5.21: Customize Screen

The user can also customize his or her page with ease. They simply double click on the text fields to change the corresponding colors and fonts. When the text field is clicked, either a color chooser dialog or a font dialog is displayed. When the user updates a field, a sample table is displayed below to show the user what its changes look like. To accept changes, the user clicks the "Accept Changes" button. If the user does not like his or her changes and wants to revert to the default color scheme, "Revert to Default" is clicked.

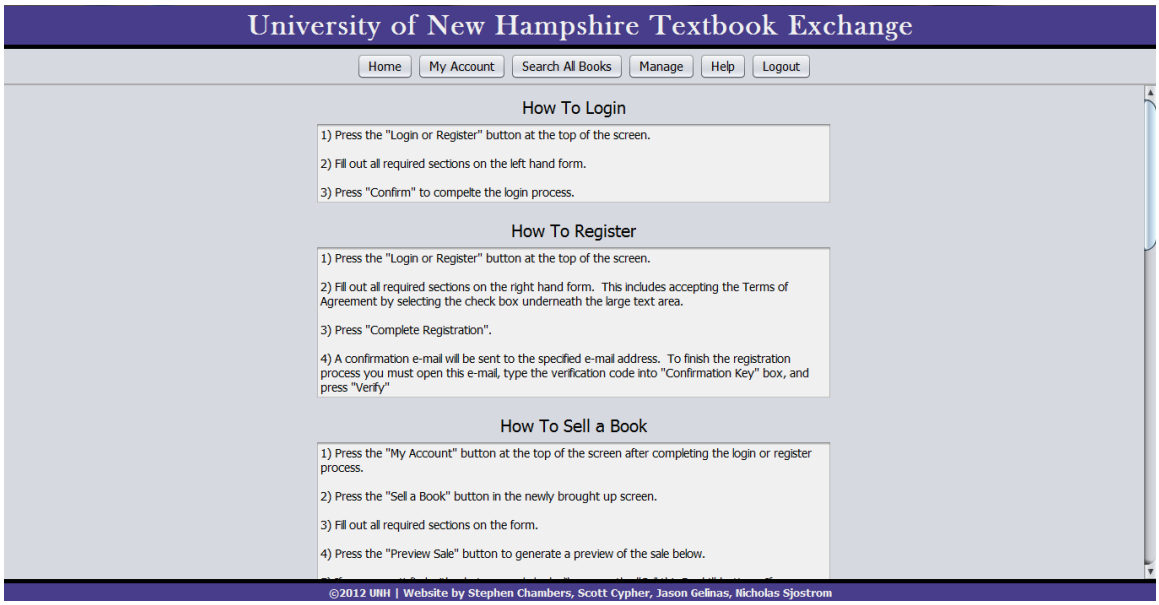


Figure 5.22: Help Screen

If at any point the user needs help in using the application, they can click the help button. The help button gives detailed descriptions on how to perform everything the user would want to in the application, from logging in to logging out.

6. Software Test Plan

All testing in the program can be run by un-commenting the instantiation of the TestSuite class in the main method of MainFrame, which is the starting point for the application. All testing was done using JUnit testing.

6.1. Correctness Testing

Correctness testing was done using white box testing. This approach allowed the interaction of objects within the application to be extensively tested.

ServerAccessPointTest

Handles SQL requests and queries

1. Connect**:** Verifies that the application is able to connect to the remote SQL server using the JDBC driver.

2. Disconnect: Verifies that the application is able to disconnect from the server.
   Add to User Information: Verifies that the server is able to successfully add a user account into the "UserInformation" table of the SQL database. Validates that all user fields can then be retrieved by the application once the user is stored. Also validates that the user can be successfully deleted.

3. Add to Book Information: Verifies that the server is able to successfully add a book into the "BookInformation" table of the SQL database. Validates that all book fields can then be retrieved by the application once the book is stored. Also validates that the book can be successfully deleted.

4. Modify Information: Verifies that the application is able to successfully modify user and book information.

5. PrimaryKey: Verifies the server is able to choose the correct primary key field based on table name.

BookRequestInfo

Handles fetches to Amazon for book information

1. Retrieve Images: Verifies that the application can retrieve a small, medium, and large image from Amazon using the Amazon Product API when given a valid ISBN number.

2. Retrieve Item Attributes: Verifies that the application can retrieve the author, title, binding, ISBN13, ISBN, and language from Amazon using the Amazon Product API when given a valid ISBN number.

3. BookTest: Encapsulates a book

4. Accessors: Verifies that accessors within book return the correct values.

5. Calculate Primary Key: Determines that the primary key for books sold is calculated correctly. The primary key calculation is as follows.
    i. Append the first letter of the seller of the book in uppercase format.
    ii. Append the first letter of the title of the book in uppercase format.
    iii. Append the first letter of the author of the book in uppercase format.
    iv. Append the first letter of the condition of the book in uppercase format.
    v. Increase a static integer within ServerAccessPoint and append that integer to the end of the primary key.

UserTest

Encapsulates a user

1. Accessors: Verifies that accessors within user return the correct values.

RegisterUserTest

1. Register a User: Verifies the functionality of the "Register User" use case.
2. Ensures that incorrect information cannot be entered into text fields.
3. Ensures that the user agreement check box must be pressed in order to complete the registration process.

SellBookTest

1. Sell a Book Verifies the functionality of the "Sell Book" use case.
2. Checks if invalid ISBN numbers cause problems with the program.
3. Checks if invalid prices cause problems with the preview book panel.
4. Ensures that a BookDisplayPanel is successfully retrieved when given valid information
5. Ensures that the book information has been properly added to the Book Information table on the *mySQL* server after confirming the sale.

SearchBookTest

1. Search a Book: Verifies the functionality of the "Search Book" use case.
2. Checks to see if queries that yield no book results don't crash the problem and properly update the screen notifying the user a book could not be found.
3. Checks to see if the table is properly updated with all values that were retrieved from the *mySQL* server

6.2. Performance Testing

1. Sending multiple queries to the SQL server in rapid succession.

   This testing is done through rapidly querying the server while searching for books. Application does not crash when search info button is pressed in rapid succession.

2. Sending rapid fetch requests to Amazon.

   Application does not crash due to hash table implementation feature. Whenever a book is queried, all of its information is stored in a hash table object for easy access later on in the program.

6.3. User Interface Testing

Due to the complexity of the system, the user interface testing is currently done by hand. The application was run by several users in order to achieve satisfactory interface flow.

Interface Flow

1. The application is able to, when run, flow through all of the different user interface screens successfully.

Status

1. The application only allows buttons to be shown based on status
   i. Only logged in users can see the “Search all Books” button
   ii. Only administrators can see the “Manage” button)

Amazon Display

1. Requests from Amazon are successfully displayed when a table with a valid ISBN stored is clicked.
2. Multiple Amazon requests are handled correctly, regardless of how many are sent.

Customization Display

1. After being customized, all tables generated display based on the values specified by the user in customization screen.
2. Customization holds true even after exiting and restarting the application.

6.4. Robustness Testing

Robustness testing was based on how the application handled invalid input. Regular expressions, class API, and multiple other testing methods are used. Robustness testing was improved over the last iterations by checking all fields at once rather than separately. This allowed the user to get more feedback and increased the general flow of the application. Additionally, Robustness testing has expanded to fit the following categories.

1. Validating User Input
2. Malicious use prevention

Validating User Input

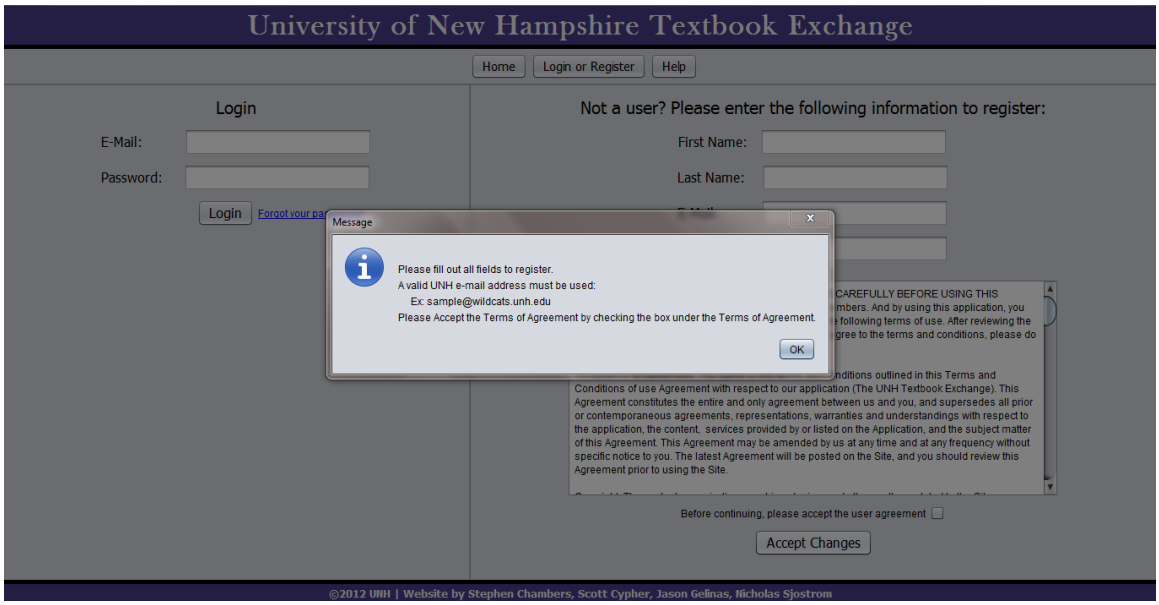


Figure 6.4.1 All-at-once input check

Checks to see if the application can successfully handle all and empty input.

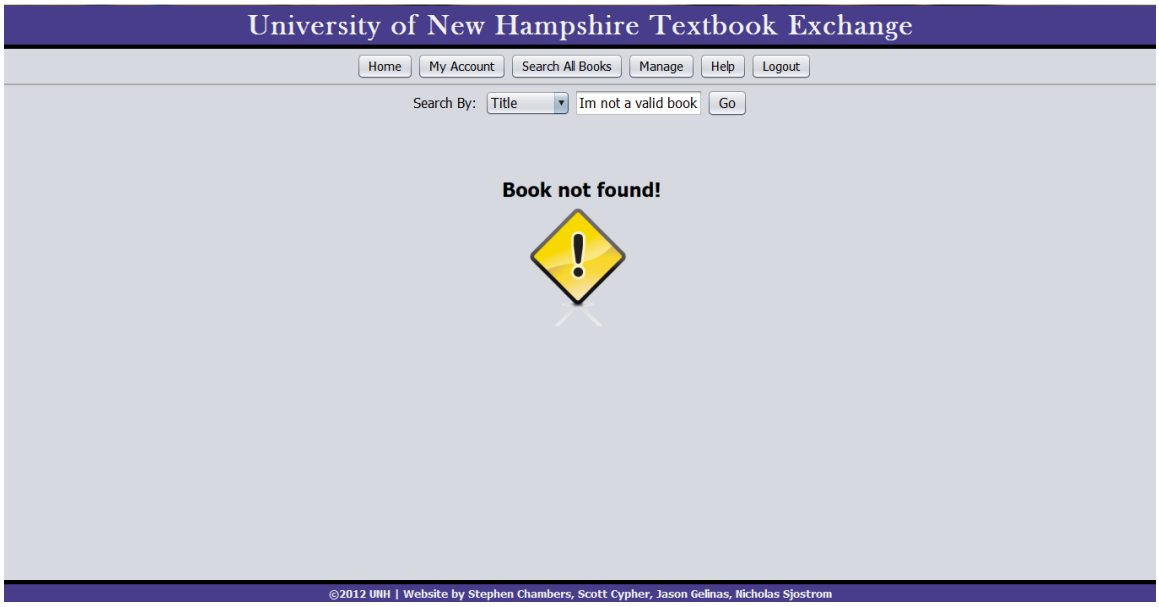


Figure 6.4.2. Book not found

Checks to see if the application successfully handles a query for a non-existent book.

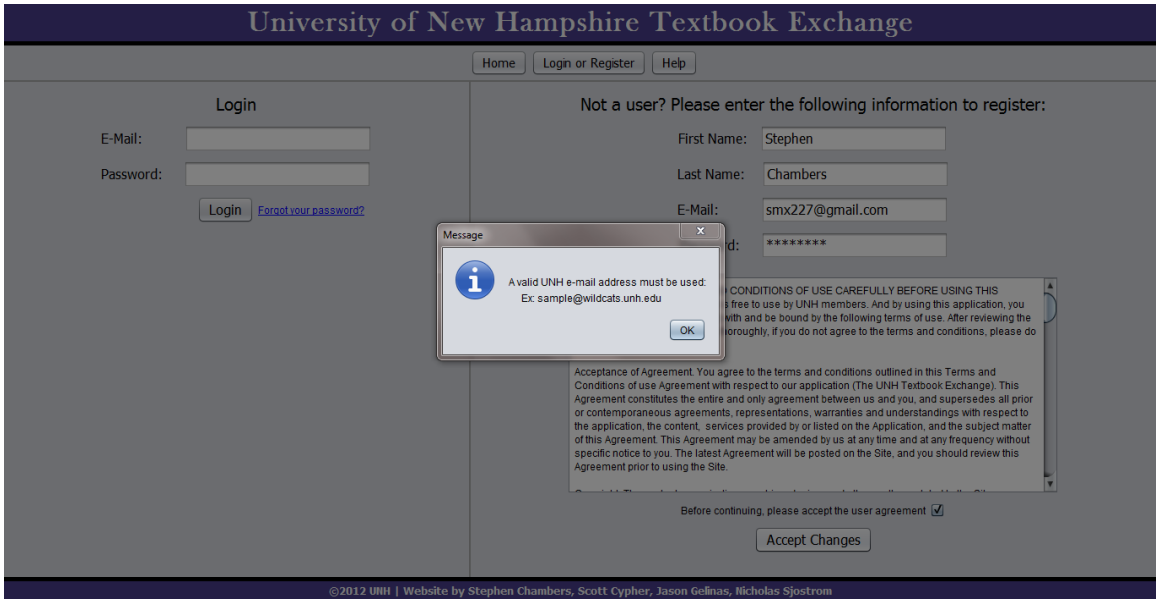


Figure 6.4.3. Invalid E-Mail

Verifies that the user entered a valid email. Checked with a regular expression: *"..*@.*[unh.edu]"*

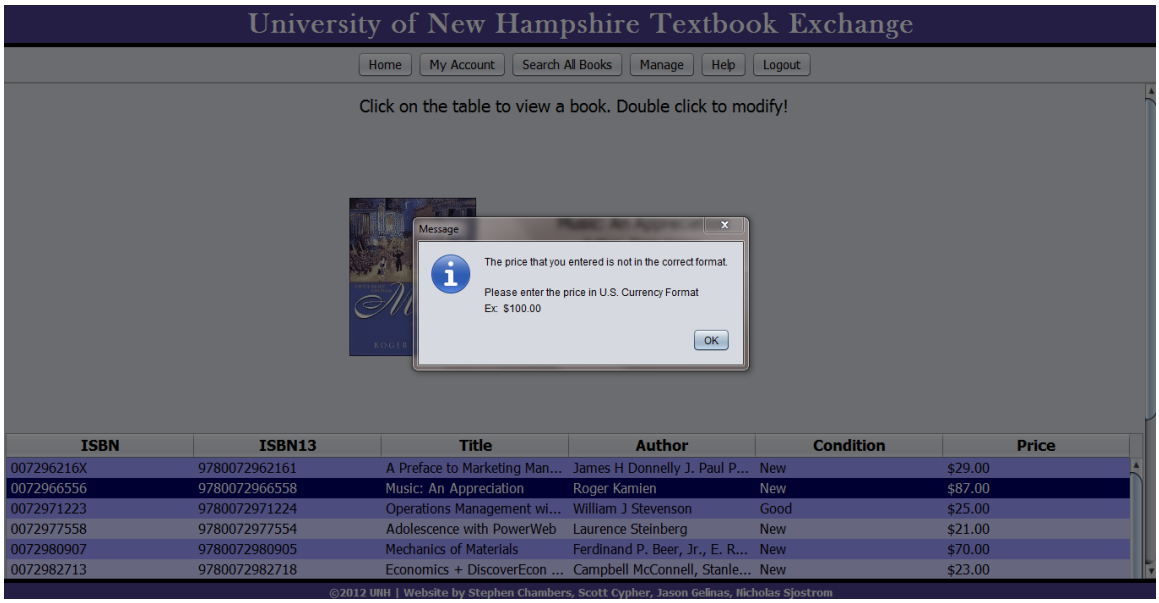


Figure 6.4.4. Invalid Book Price

Verifies that the user entered a valid price. Checked with the regular expression:

"[$]([0-9][0-9]?([,][0-9]{3}){0,4}([.][0-9]{0,4})?)$|^[$]?([0-9]{1,14})?([.][0-9]{1,4})$|^[$]?[0-9]{1,14}$"

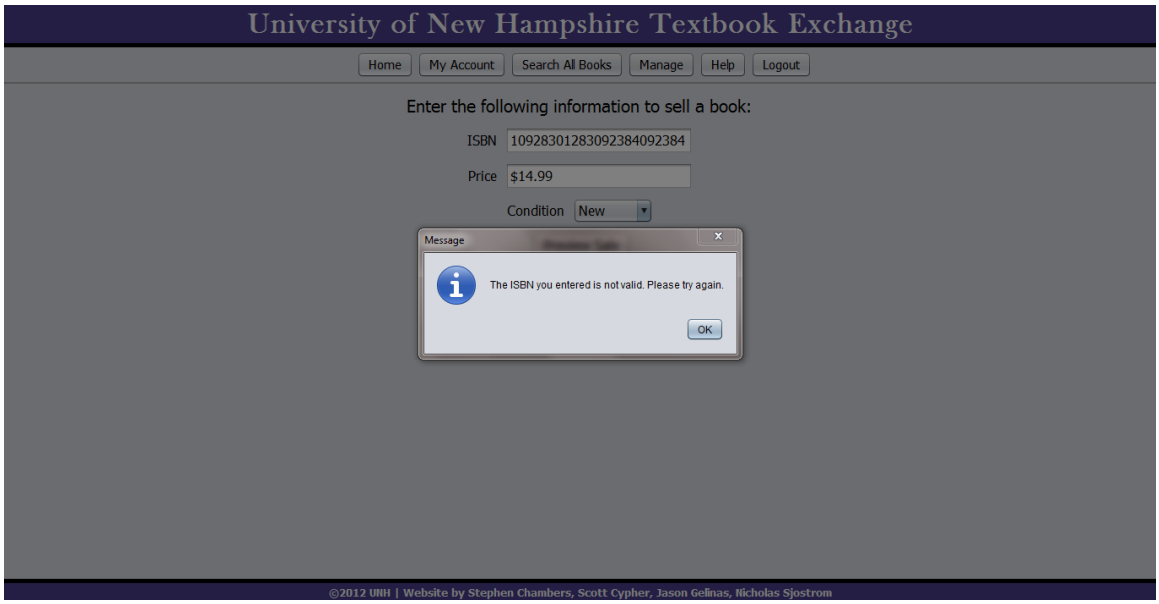


Figure 6.4.5. Invalid ISBN

Verifies that the user entered a valid ISBN. Checked through the regular expression *"^(97(8|9))?[0-9]{9}([0-9]|X)$" and by testing the results from the Amazon server.*

Malicious Use Prevention

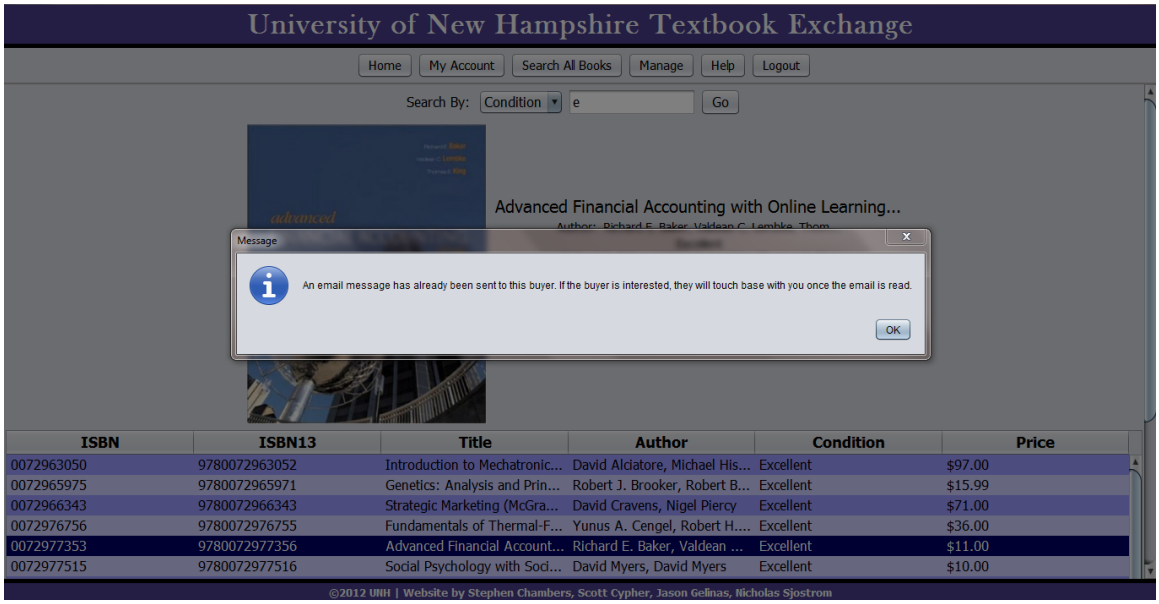


Figure 6.4.6. Spam

Testing was done to ensure the user was limited to the number of requests they could send for a book.

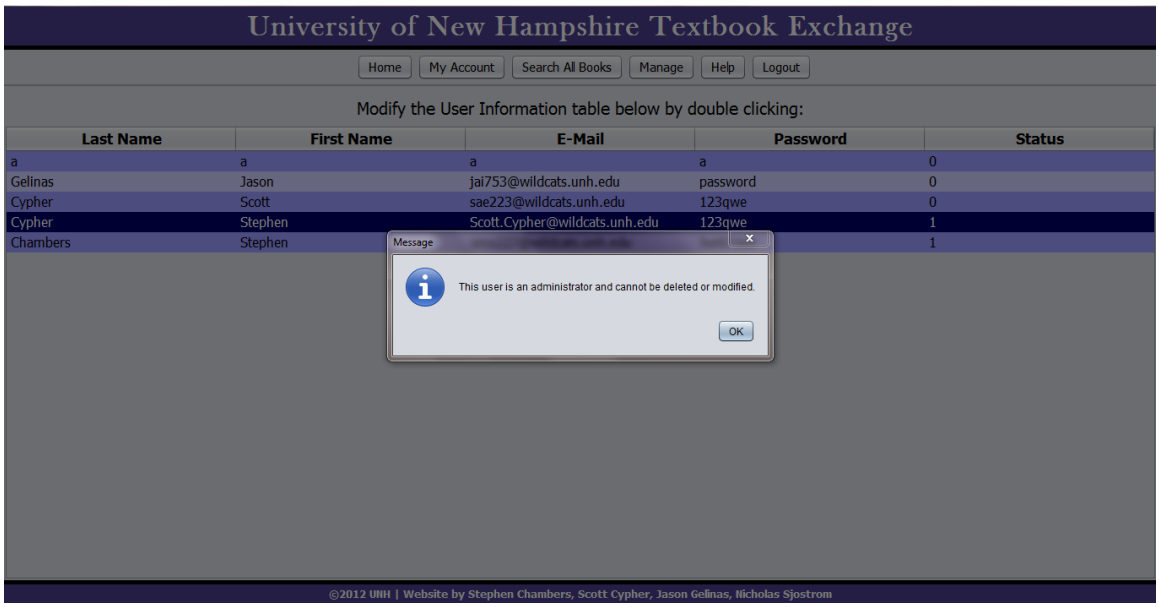


Figure 6.4.7. Administrative Power Abuse

Testing was done to ensure a single administrative user could not take over the system.

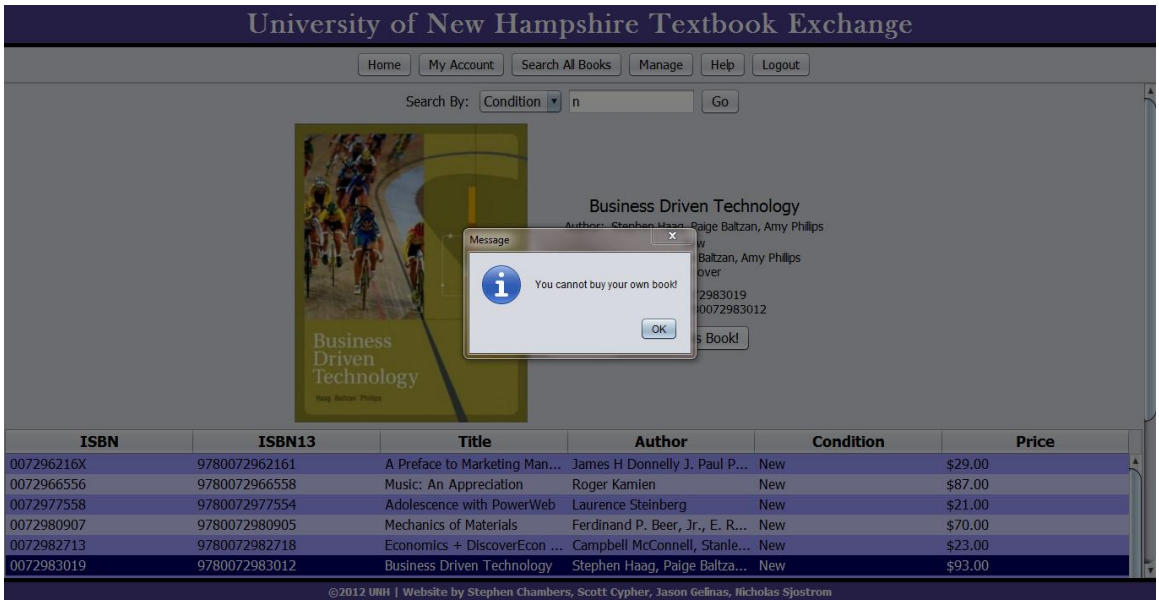


Figure 6.4.8. Cannot buy own book

Testing was done to ensure the user was not capable of buying their own books from the server.

7. UML Tool Documentation

The UML tool used to create UML diagrams for this project was UMLet. UMLet is a free Eclipse plug-in that is capable of quickly generating UML diagrams. This tool allowed us to easily generate use case diagrams, sequence diagrams, and UML diagrams that could be simply pasted into the report.

Upon opening UMLLet, the user is met with a frame containing 3 predominant regions: the editor region, components region, and the text editing region. The editor region is where the diagram currently being manipulated can be located. The components region is where the UML diagram components are located. This includes relational operators, inheritance operators, reference operators, package components, and class components. The text editing region is the area used to modify the text of a component. By simply placing a component into the editor region and typing information into the text editing region, UML diagrams can be quickly created, updated, and customized.

8. References

"9 Qualities of Good Software | Worthwhile.com." *Worthwhile.com*. N.p., n.d. Web. 08 Nov. 2012. <http://www.worthwhile.com/blog/what-makes-software-worthwhile/>.

"About Java and JRuby Development JEE, Spring, Guice Hibernate, Java Persistence (JPA) and Various Web Frameworks." *Eclipse Junit Testing Tutorial*. N.p., n.d. Web. 03 Nov. 2012. <http://www.laliluna.de/articles/posts/eclipse-junit-testing-tutorial.html>.

"Accessing Amazon Product Advertising API." *Accessing Amazon Product Advertising API*. N.p., n.d. Web. 23 Nov. 2012. <http://www.codediesel.com/php/accessing-amazon-product-advertising-api-in-php/>.

"Black Box Testing." *Software Testing Fundamentals*. N.p., n.d. Web. 19 Oct. 2012. <http://softwaretestingfundamentals.com/black-box-testing/>.

"Black Box Testing." *Webopedia*. N.p., n.d. Web. 04 Nov. 2012. <http://www.webopedia.com/TERM/B/Black_Box_Testing.html>.

"Creating UML Use Case Diagrams." - *Developer.com*. N.p., n.d. Web. 27 Oct. 2012. <http://www.developer.com/design/article.php/2109801/Creating-Use-Case-Diagrams.htm>.

"Design Patterns." *Design Patterns*. N.p., n.d. Web. 22 Oct. 2012. <http://sourcemaking.com/design_patterns>.

"Design Patterns in Java(TM) (Software Patterns Series) [Hardcover]." *Design Patterns in Java(TM) (Software Patterns Series): Steven John Metsker, William C. Wake:*

9780321333025: Amazon.com: Books. N.p., n.d. Web. 22 Oct. 2012. <http://www.amazon.com/Design-Patterns-Java-TM-Software/dp/0321333020>.

"Developing Java Software." *:*. N.p., n.d. Web. 19 Oct. 2012. <http://www.devjavasoft.org/>.

"Developing Java Software (third Edition) [Paperback]." *Developing Java Software (third Edition): Russel Winder, Graham Roberts: 9780470090251: Amazon.com: Books*. N.p., n.d. Web. 19 Oct. 2012. <http://www.amazon.com/Developing-Software-third-Russel-Winder/dp/0470090251>.

"Domain Model Diagram." *C Board*. N.p., n.d. Web. 03 Nov. 2012. <http://cboard.cprogramming.com/tech-board/141336-domain-model-diagram.html>.

"Domain Model." *P of EAA:*. N.p., n.d. Web. 03 Nov. 2012. <http://martinfowler.com/eaaCatalog/domainModel.html>.

"Help - Eclipse Platform." *Help - Eclipse Platform*. N.p., n.d. Web. 08 Nov. 2012. <http://help.eclipse.org/juno/index.jsp?topic=/org.eclipse.jdt.doc.user/gettingStarted/qs-junit.htm>.

"Introduction to UML 2 Sequence Diagrams." *Introduction to UML 2 Sequence Diagrams*. N.p., n.d. Web. 26 Nov. 2012. <http://www.agilemodeling.com/artifacts/sequenceDiagram.htm>.

"Introduction To UML 2 Use Case Diagrams." *Introduction To UML 2 Use Case Diagrams*. N.p., n.d. Web. 03 Nov. 2012. <http://www.agilemodeling.com/artifacts/useCaseDiagram.htm>.

"Java 2 Platform SE 5.0." *Java 2 Platform SE 5.0*. N.p., n.d. Web. 27 Oct. 2012. <http://docs.oracle.com/javase/1.5.0/docs/api/>.

"Java Design Patterns." *Java Design Patterns*. N.p., n.d. Web. 30 Oct. 2012. <http://www.javaworld.com/columns/jw-java-design-patterns-index.html>.

"Java Platform SE 6." *Java Platform SE 6*. N.p., n.d. Web. 05 Nov. 2012. <http://docs.oracle.com/javase/6/docs/api/>.

"Java Software Development." *J2EE Web Development Java Programmers Hiring Java Consultants*. N.p., n.d. Web. 27 Oct. 2012. <http://rndinfo.com/java-software-application-development.html>.

"Java Swing Tutorial." *Java Swing Tutorial*. N.p., n.d. Web. 27 Oct. 2012. <http://zetcode.com/tutorials/javaswingtutorial/>.

"Java SwingÂ -Â O'Reilly Media." *Java SwingÂ -Â O'Reilly Media*. N.p., n.d. Web. 30 Oct. 2012. <http://shop.oreilly.com/product/9781565924550.do>.

"Java Virtual Machine." *Java Virtual Machine*. N.p., n.d. Web. 03 Nov. 2012. <http://www.helpbytes.co.uk/java.php>.

"JUnit." *JUnit*. N.p., n.d. Web. 09 Oct. 2012. <http://junit.sourceforge.net/>.

"JUnit Testing Utility Tutorial." *JUnit Testing Utility Tutorial*. N.p., n.d. Web. 19 Oct. 2012. <http://supportweb.cs.bham.ac.uk/documentation/tutorials/docsystem/build/tutorials/junit/junit.html>.

"JVM." *Webopedia*. N.p., n.d. Web. 19 Oct. 2012. <http://www.webopedia.com/TERM/J/JVM.html>.

"Microsoft Java Virtual Machine Support." *Microsoft Java Virtual Machine Support*. N.p., n.d. Web. 04 Nov. 2012. <http://www.microsoft.com/About/Legal/EN/US/Interoperability/Java/Default.aspx>.

"MySQL :: The World's Most Popular Open Source Database." *MySQL :: The World's Most Popular Open Source Database*. N.p., n.d. Web. 19 Oct. 2012. <http://www.mysql.com/>.

"MySQL Documentation: MySQL Reference Manuals." *MySQL ::*. N.p., n.d. Web. 03 Nov. 2012. <http://dev.mysql.com/doc/>.

"PHP MySQL Introduction." *PHP MySQL Introduction*. N.p., n.d. Web. 03 Nov. 2012. <http://www.w3schools.com/php/php_mysql_intro.asp>.

"Product Advertising API." *Product Advertising API*. N.p., n.d. Web. 29 Oct. 2012. <https://affiliate-program.amazon.com/gp/advertising/api/detail/main.html>.

"Sample Code & Libraries." *: Amazon Web Services*. N.p., n.d. Web. 30 Oct. 2012. <http://aws.amazon.com/code/Product-Advertising-API>.

"Swing(JavaTM Foundation Classes)." *JDK 6 Swing (Java Foundation Classes (JFC))-related APIs and Developer Guides*. N.p., n.d. Web. 19 Oct. 2012. <http://docs.oracle.com/javase/6/docs/technotes/guides/swing/>.

"Trail: Creating a GUI With JFC/Swing." *(The Javaâ„¢ Tutorials)*. N.p., n.d. Web. 02 Nov. 2012. <http://docs.oracle.com/javase/tutorial/uiswing/>.

"Tutorial 29: Black Box Testing." *Tutorial 29: Black Box Testing*. N.p., n.d. Web. 29 Oct. 2012. <http://www.guru99.com/black-box-testing.html>.

"UML Basics: The Sequence Diagram." *UML Basics: The Sequence Diagram*. N.p., n.d. Web. 27 Oct. 2012. <http://www.ibm.com/developerworks/rational/library/3101.html>.

"UML Sequence Diagrams." *UML Sequence Diagram*. N.p., n.d. Web. 03 Nov. 2012. <http://www.altova.com/umodel/sequence-diagrams.html>.

"UML Use Case Diagrams: Tips." *UML Use Case Diagrams: Tips*. N.p., n.d. Web. 03 Nov. 2012. <http://www.andrew.cmu.edu/course/90-754/umlucdfaq.html>.

"Unit Testing Your Application with JUnit." *Unit Testing Your Application with JUnit*. N.p., n.d. Web. 03 Nov. 2012. <http://www.oracle.com/technetwork/articles/adf/part5-083468.html>.

"Welcome to JavaWorld.com." *Welcome to JavaWorld.com*. N.p., n.d. Web. 29 Oct. 2012. <http://www.javaworld.com/>.

"What Is Software Quality?" *What Is Software Quality?* N.p., n.d. Web. 20 Oct. 2012. <http://www.ocoudert.com/blog/2011/04/09/what-is-software-quality/>.