VERSION 1.4

NOVEMBER 24, 2016

DESKTOP APPLICATION BOOKSTORE

REQUIREMENTS ELICITATION

CSC4350 SOFTWARE ENGINEERING (FALL 2016)

PRESENTED BY: THE CODE MONSTERS

DINO CAJIC | ZACHARY SHOULTS | ANH PHAM

CONTENTS

SECTION 1: PROJECT DESIGN

1.1: TEAM ROLE STRUCTURE

Team member name	Primary Role	Secondary Role	Current Status
Dino Cajic	Project Manager	Client	Active
Zachary Shoults	Technical Writer	Developer	Active
Anh Pham	Developer	Client	Active
Piero Carruitero	Human Factors Specialist	User	Inactive
William Chiu	N/A	N/A	Inactive
Jerry Vu	N/A	N/A	Inactive

1.2: TEAM ROLES AND RESPONSIBILITIES

Dino Cajic
Primary: Project Manager | Secondary: Developer | Tertiary: Technical Writer

As an elected project manager of the team, my responsibilities will be to manage the team and project scope, help with any tasks throughout the project, and handle any issues that may arise. The amount of time spent on the scope of the project usually dictates how well and how soon the overall project is understood within the team and by the customer; I will be devoting a significant amount of time in the beginning to make sure that the scope is understood. My other responsibility is to coordinate all of the resources and tasks. The work will need to be done in sequence with minimum time wasted. To accomplish this, I will need to facilitate communication between the members of the team. When project issues arise, the team can rely on me to handle them since I will be the main point of contact that oversees every aspect of the project.

My secondary role will be a developer. I create the entire structure, populate the necessary content required for the team to start filling in the necessary code and help out the team by creating the code myself.

My third role will be to help with the technical writing. The Technical Writer details will work with the main Technical Writer (Zachary Shoults) to create the necessary documentation. I'm expected to write certain portions of the document but will concentrate mainly on the look of the document. I'm expected to create all of the diagrams within the documents.

Zachary Shoults
Primary: Technical Writer | Secondary: Developer

Technical writers have a full grasp of all users' possible interactions with the final product. I expect to write a comprehensive document detailing the functionality of the software. Any user should be able to rely on this manual to step through a full session using the program and accomplish any goal the program promises to deliver (i.e. purchase an item, add an item to stock, change product information,

query the inventory). My secondary role will be as developer to create the software and interface. An understanding of the coding and implementation of the product will improve my understanding of the product's internal functionality and therefore make the manual easier to write.

Anh Pham
Primary: Tester | Secondary: Client

My main role will be to test the software once the developers create it. Once each class has been thoroughly tested, I will report the findings to the team.

My role has evolved to help assist the Technical Writers with whatever they may need.

Piero Carruitero *(Inactive)*
Primary: Human Factors Specialist | Secondary: User

In the role of human factors specialist, my input will drive product design and performance to optimize users' experiences. I want to ensure that the final software is presented to users with simple and intuitive, yet powerful, features. The user interface should not require any outside instruction to explain how to accomplish the most common use-cases. User's should not feel intimidated by the program; they should feel like it is a powerful and simple tool that they should have thought of. My secondary role, fittingly, will be that of a user. This role will help me focus on the application domain early in the project's life-cycle, then I will utilize those insights to satisfy my responsibilities as a human factors specialist.

Considering my experience in UI/UX, I will be expected to provide input for any nonfunctional requirements that may arise related to the look and feel of the software. This is especially crucial during the Requirements Elicitation stage of the project.

1.3: PROJECT DATES

Document	Start Date	Due Date	Status
Project Contract	08/27/2016	09/03/2016	Completed
Requirement Elicitation	09/03/2016	09/24/2016	Completed
System Analysis	09/24/2016	10/01/2016	Completed
Object Design	10/01/2016	10/22/2016	Completed
Project Rational	10/22/2016	10/29/2016	Completed
Testing Document	10/29/2016	11/05/2016	Completed
Final Document	08/27/2016	11/19/2016	Completed

1.4: GANTT CHART

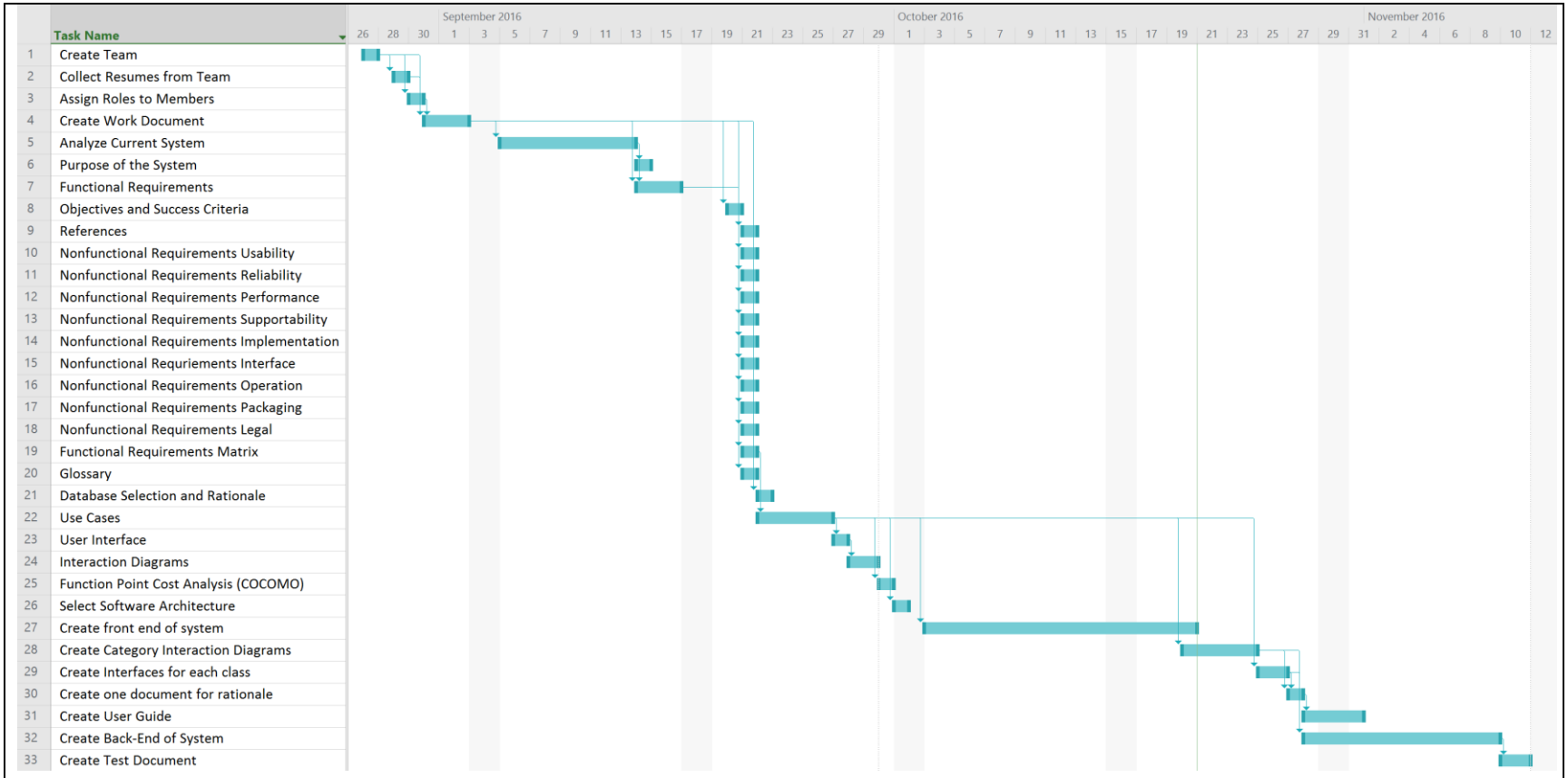

Figure 1: Gantt Chart displaying the current development progress of the Bookstore Desktop Application.

SECTION 2: INTRODUCTION

2.1: PURPOSE OF THE SYSTEM

The system proposed is an all-in-one system solution for a small-business that needs to unify their inventory, sales and expenses into one software application. The system will also provide a modern experience for the customer by allowing them to research books from any of the available computers running the software; these books are the books located in the bookstore. The main purpose of the system is to increase efficiency and eliminate human-errors that have arisen through the company's expansion.

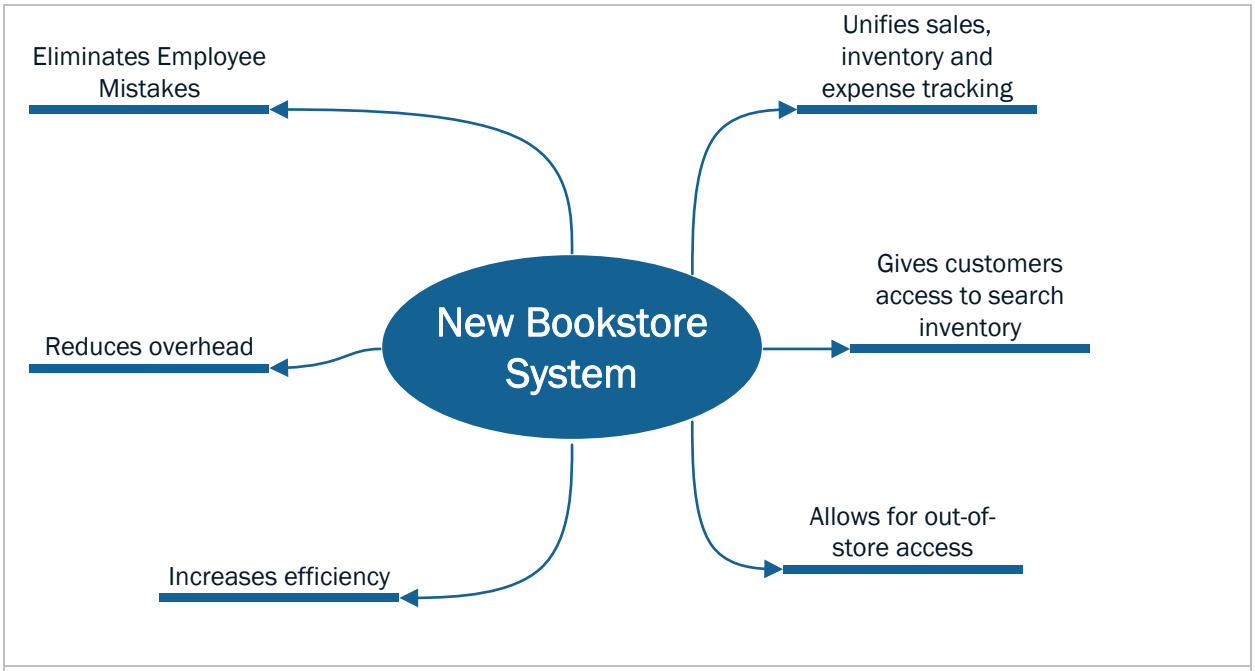

Figure 2: What the Bookstore system hopes to achieve

2.2: SCOPE OF THE SYSTEM

The Bookstore will upgrade from its current system to the new system by November 23, 2016. This project will replace the existing Point of Sales system, inventory management system, reporting system and expense tracking system. It will also include a new interactive system allowing for customers in the bookstore to browse the in-stock inventory. The data is to be kept offsite and a redundant backup-management system in place is to be kept within the bookstore.

2.3: OBJECTIVES AND SUCCESS CRITERIA OF THE PROJECT

In order for the software application to be accepted by the customer, Code Monsters will have to deliver the project in its entirety; the crucial requirements are listed in Section 4.2, Functional Requirements. As was agreed upon with the customer, it is mandatory that November 29th, 2016 be the launch date of the new software; November 29th marks the beginning of the new fiscal year for the customer. Prior to the launch, all necessary personnel training needs to be completed. A member from the Code Monster staff needs to spend at minimum one day with the staff immediately following the successful launch in case of any major issues that may arise.

2.4: REFERENCES

Existing Systems the customer wanted us to reference:

http://basilsoftware.com/

http://www.anthology.com/anthologyweb/default.aspx

https://www.shopify.com/pos/retail/bookstore

http://www.booklog.com/

https://www.cashierlive.com/book-store-pos.html

http://www.retailpoint.com/retail-pos/book-store-pos/

2.5: OVERVIEW

The document depicts the functional and nonfunctional requirements of the bookstore desktop application. The current system is described in detail and the proposed system is reviewed. Upon successful completion and acceptance of the Requirements Elicitation Document, the Analysis Document will be created and appended to Section 4.5 of this document.

SECTION 3: CURRENT SYSTEM

The Bookstore program is being designed for a Bookstore retail brick-and-mortar store. The current system in place for keeping inventory is an Excel sheet. Receipts from the register are gathered at the end of the day and the manager will enter the sales into the main Excel sheet. A summary sheet retrieves data from the main sales sheet and helps create the daily and monthly sales reports through certain Excel formulas. The inventory sheet will retrieve data from the sales sheet, mainly the ISBN and the quantity sold, and either add or subtract inventory from the item that has the same ISBN.

The bookstore utilizes an employee excel sheet to track employee expenses, and another sheet to track all other expenses. These include: electricity, gas, rent, phone, internet, water and miscellaneous (for example shelving for the books or a new register). Payments are processed through First Data's in-store terminal and each payment is sent to their SunTrust bank account. Data flow between excel sheets can be seen in Figure 3.

Issues with the current system:

1. Management staff has to enter each sale into the Main Sales Excel Sheet. As the company continues to grow, this portion is taking a significant amount of time.
2. There are a number of mistakes that the management keeps making when entering each sale into the Main Sales Excel Sheet. These include:
    A. Entering the wrong ISBN
    B. Entering the wrong quantity
    C. Entering the wrong price that the item was sold for
3. Management has too much control. There is nothing stopping the management from going into the system and altering prices and quantities. The owner is worried about theft.
4. The system doesn't automatically generate the reports; the company wants one system that integrates expenses, sales and inventory and automatically generates reports.
5. There's no way for the customers to browse for books digitally.
6. Access to the inventory and sales has to be done within the store alienating the possibility of outside sales selling books at events.

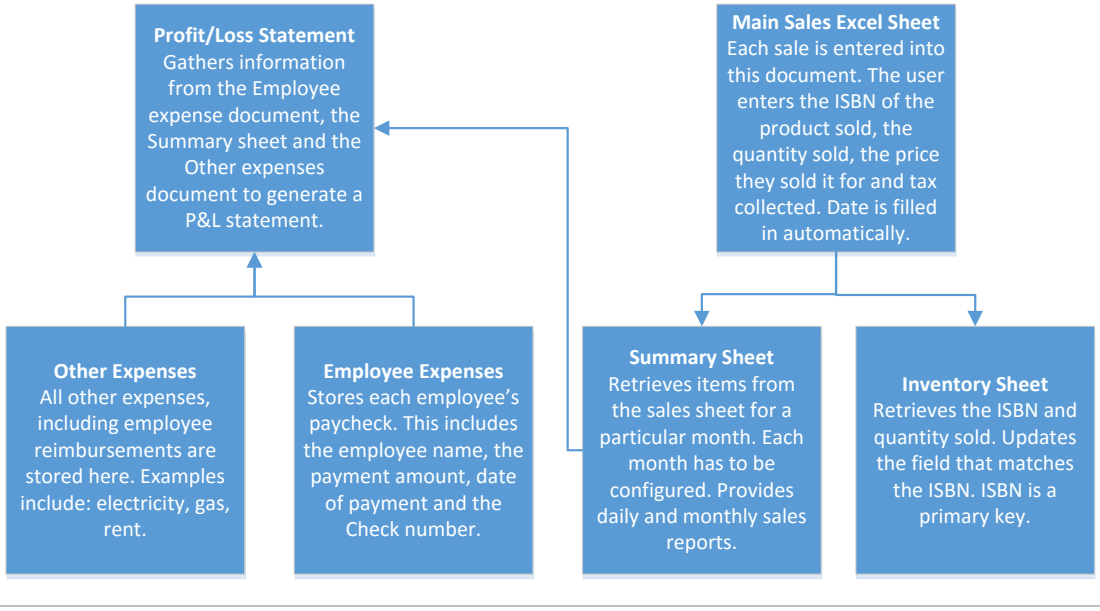

Figure 3: Current System Flow

SECTION 4: PROPOSED SYSTEM

4.1 OVERVIEW

The Proposed System will be a desktop application servicing the specific needs of a typical bookstore. The system is to be stripped down of unnecessary clutter that other systems are consumed with since efficiency and usability are key factors for the customer. The system will provide on-site and off-site access; each terminal will connect to the server that's stored off-site. Customers shall have the capability to browse the current inventory. In addition to all of the features that the typical customer has, administrators and employees shall have the ability to process sales, create stocking orders and generate profit loss reports. The full functionality of the proposed bookstore desktop application can be reviewed in Section 4.2, Functional Requirements.

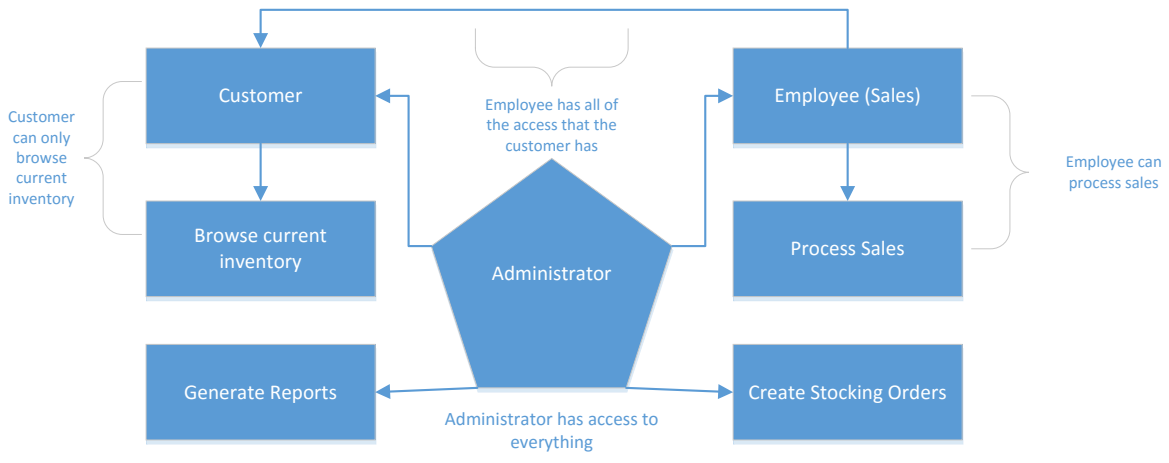

Figure 4: Software Accessibility

A thorough breakdown of the nonfunctional requirements, detailing the usability, reliability, performance, supportability, implementation, interface, operation, packaging, and legal requirements can be reviewed in Section 4.3.

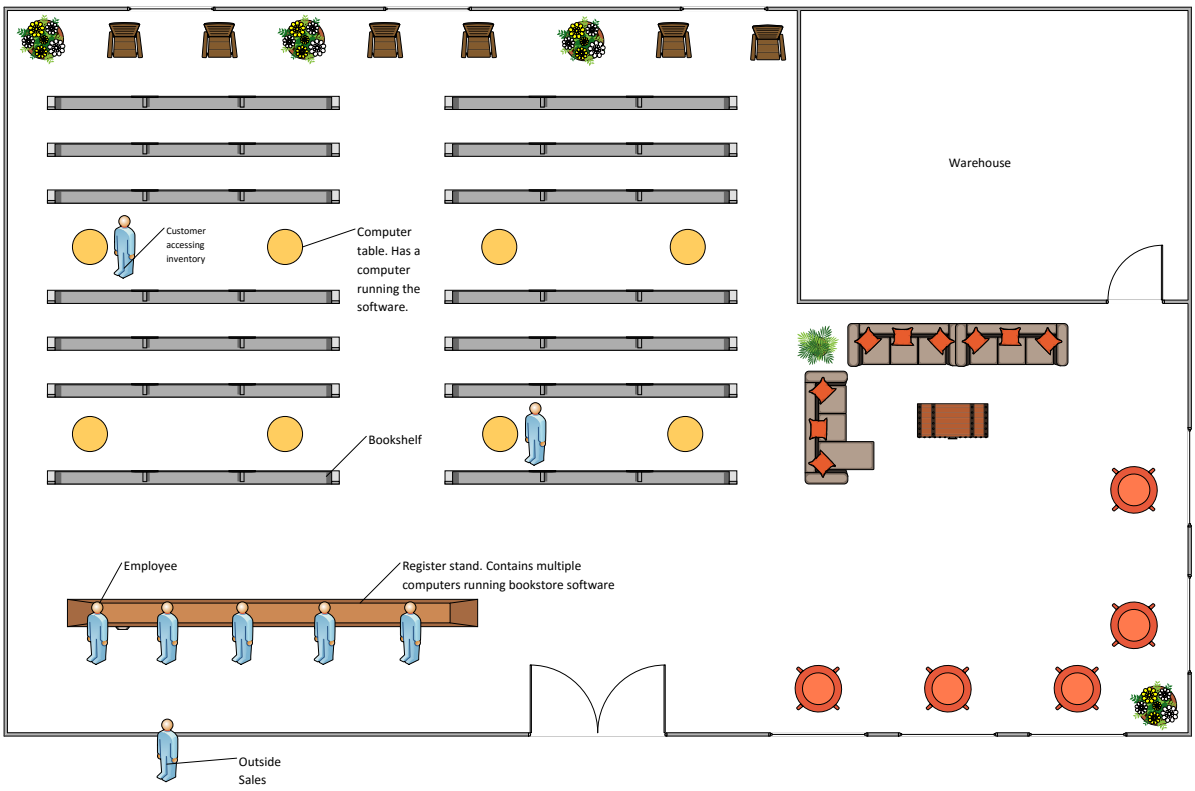

Figure 5: Bookstore Layout. Shows types of interactions different users might have with the application.

4.2 FUNCTIONAL REQUIREMENTS

Code Monsters will be developing a book inventory system that shall be accessible to both customers and staff at the book store's location. The system shall be a software package that installs on dedicated computers throughout the store providing an interface for the user to see the state of the inventory at any time. These terminals shall work independently of one another. The system shall reconcile any inventory changes made from a terminal by each referencing a consolidated, current record of the inventory maintained on an off-site server via the internet. Any terminal, at any given time, shall provide the same inventory information to the user because they all query against the same master-copy of the inventory.

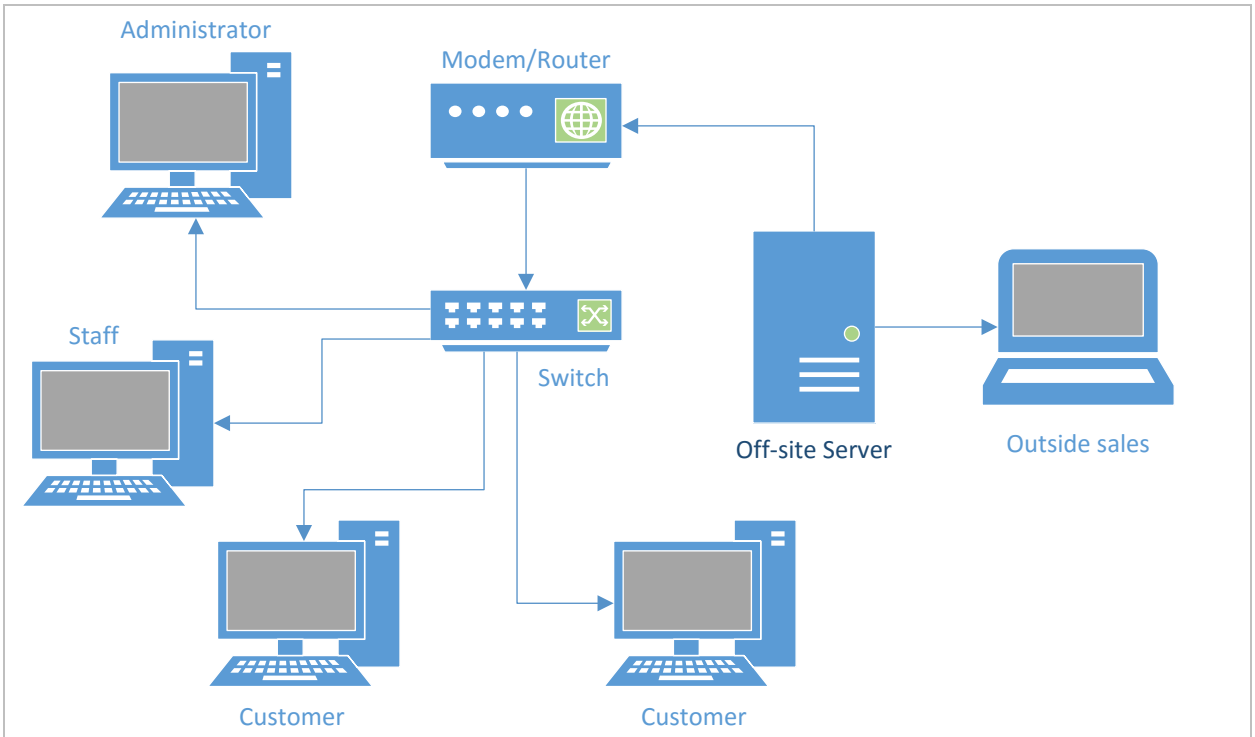

Figure 6: Network Diagram for Bookstore. Shows internal and external server access.

The system shall track every book the store sells. The inventory record for every book shall contain the book's ISBN, title, author(s), genre, summary, price, cost, and stock currently on-hand. The system shall track every book the store orders. Orders shall be identified by their order number and they should contain every book's ISBN, title, author(s), genre, summary, price, cost, quantity on-order, and expected date of arrival. The data for the books on-order shall be structured in the same way as the data for the books on the in-store inventory so that the interfaces for searching either list shall be the same. Every book shall have its own page on the system to show all information available about the book. Search results for books shall only show a book's title, author(s), genre, price, and stock quantity to allow more results to be displayed at once.

Searching for inventory items shall, by default, use a book's title. Options to search by one or multiple of the following categories shall be available to the user: title, ISBN, author, keywords, or genre. Search results shall be displayed with perfect-matches first, followed by any other valid results, all in ascending alphabetical order. Results that have two or less copies remaining in-stock shall show the stock quantity in an italicized, bold, or highlighted font. If a search yields a result that has a stock quantity of zero, the system shall automatically check if that book is in an order. If that book is on-order, the soonest-expected restocking date shall be displayed to the user with the results of the original inventory search. An option to browse the store's inventory by book's category, shall be available to all users.

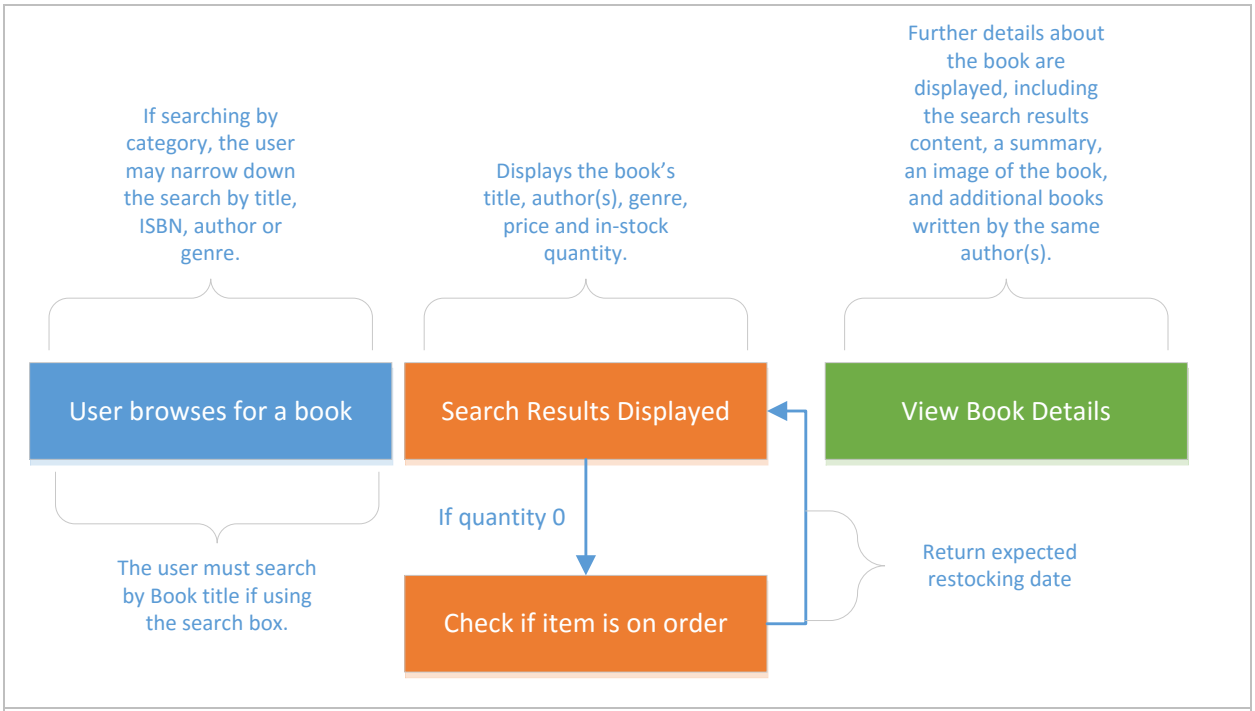

Figure 7: Flow diagram depicting a typical search using the Bookstore application

Searching existing orders shall only be explicitly performed by registered users (staff or administrator users). An order search shall be done using one or multiple of the following categories: title, ISBN, author, keywords, genre, order number, or delivery date. Search results shall be displayed with perfect-matches first, followed by any other valid results, all in order by delivery date with the most recent date first. Once delivered, an order shall be added to the inventory in the system by a registered user. By default, a registered user shall be able to add an entire order to the inventory. This shall add each book on that order to the inventory. If that book was already on the inventory listing, the system shall update the inventory stock quantity by the amount on the order. If that book is not in the inventory, the book shall be added to the inventory using the book data from the order. In the event that not every book that was ordered was received by the store, the registered user shall be able to change the quantity of books received to reflect the actual number of books delivered to the store before updating the inventory levels. Adding a book to the inventory, without it being ordered, shall be a feature only available to administrators. Modifying a book's details in the inventory, shall also only be performed by an administrator. Likewise, removing a book entirely from the inventory shall only be done by administrators.

The system shall begin with one root administrator user. This user shall be able to create other administrator users or staff users. Any administrator user shall be able to create or delete staff users. Administrator user accounts shall only be deleted by the root administrator user. The root administrator account cannot be deleted. All registered users shall have a username and password used to log into the system. Resetting user passwords shall only be performed by administrators. Once a registered user is logged in, the actions that the particular user is allowed to perform shall be accessible. Actions that are exclusive to administrators shall not be accessible by staff users. All registered users shall be able to search existing orders, search the inventory, add books to the inventory from delivered orders, and sell items to customers. Administrator users shall be able to place new orders, create new registered user accounts, review sales information, and review system logs.

When an administrator creates a new user, the form to create the account shall have the following fields: First Name, Last Name, Employee Identifier (used as Username), and phone number. When an administrator deletes a user account, the action shall include a text field that gives a brief explanation of why the account is being deleted.

Creating new orders shall be done by any administrator. Orders shall start empty and shall be populated by using the ordering interface to search for titles or ISBNs in the inventory or existing orders. If the desired book is in the search results, the interface shall allow that book to be added to the new order with a mouse-click. If the book to be ordered has not been ordered previously, the administrator shall be able to create the entry with all the required fields filled in. Once an order is submitted, it shall receive a unique identifier that is associated with the order for future searches to reference and it shall have a status of “in-progress”. When an order is shipped and added to the inventory, its status shall be changed to “received”.

Any registered user shall be able to sell a book. Once the user logs in, the ability to initiate a sale shall become available. Sales shall require a book’s ISBN. A book shall only be sold if it’s on the inventory list. The system shall be able to execute a sale for a book even if the stock quantity for that book is zero. When a book is sold, the in-stock quantity for that item on the inventory shall be reduced by one (or left at zero if the quantity is already zero).

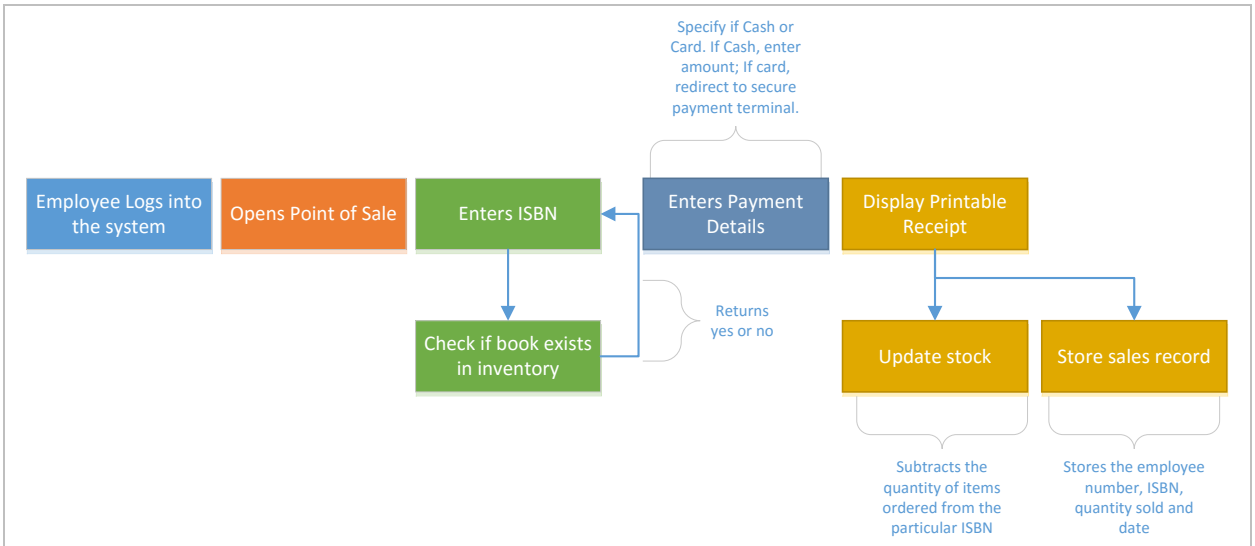

Figure 8: Flow diagram for Point of Sales

For every sale, the employee username, book details, quantity, and date shall be recorded for record-keeping. An administrator shall be able to access sales records and query them by title, ISBN, author, keywords, genre, or sale date. The sales features available to administrators shall allow the sales data to be queried using a range of consecutive dates (i.e. [March 2 2017 through March 19 2017], [October 9 2016 through present day]). In this way, the user shall be able to view all sales over a given time period. Sales older than two years shall be archived outside of the reach of the system to prevent gradually slowing down the system with data that is very rarely needed for searches.

Administrator logs shall not be searchable at initial implementation. A future goal for the system may be to track anonymous customer search query statistics, but at launch, the logs shall not need that

functionality. They shall be maintained as chronological entries where the most recent entries are at the top. Each log entry shall contain the time-stamp of the event. Logs shall track registered users' activity data. The primary purpose of the logs shall be to track staff and administrator actions on the system for accountability. Events that shall become log entries are as follows:

- Partial orders are added to inventory. The employee username that accepted the order and the item(s) that were missing shall be in the log entry.
- Administrators create a new registered user. The username that was created shall be in the log entry.
- Administrators delete a registered user. The username that was deleted and the explanation for why the user was deleted shall be in the log entry.

Administrators shall have the ability to submit statements to the logs. They shall not be able to edit the log files directly. All entries to log files shall be done through the system's interface to protect the integrity of the log files.

4.3 REQUIREMENTS TRACEABILITY MATRIX

Requirements

F: Functional **IR**: Interface **HWR**: Hardware

ID	Para	Sentence	Category	Use Case Titles	UC ID
1	18.1	The system shall be a software package … installs on dedicated computers…	**F**		
2	18.2	These terminals shall work independently of one another.	**F**		
3	18.3	The system shall reconcile any inventory changes … referencing a consolidated, current record of the inventory … on an off-site server.	**F / HWR**		
4	18.4	Any terminal … shall provide the same inventory information to the user	**F**		
5	18.4	[All terminals] … query against the same master-copy of the inventory	**F**		
6	19.0	The system shall track every book the store sells.	**F**		
7	19.1	The inventory record for every book shall contain the book's [attributes].	**F**		
8	19.2	The system shall track every book the store orders.	**F**		
9	19.3	Orders shall be identified by their order number and they should contain every book's [attributes].	**F**		
10	19.4	The data for the books on-order shall be structured in the same way as the data for the books on the in-store inventory so that the interfaces …	**F / IR**		
11	19.5	Every book shall have its own page on the system to show all information …	**F**		
12	19.6	Search results for books shall only show a book's title, author(s), …	**F**		
13	20.0	Searching for inventory items shall, by default, use a book's title.	**F**	SearchBookByTitle	**uc1**

ID	Para	Sentence	Category	Use Case Titles	UC ID
14	20.1	Options to search by one or multiple of the following categories shall be available to the user: title, ISBN, author, keywords, or genre.	**F**	SearchBookWithCustomOptions	**uc2**
15	20.2	Search results shall be displayed with perfect-matches first, followed by any other valid results, all in ascending alphabetical order.	**F**		
16	20.3	Results that have two or less copies remaining in-stock shall show the stock quantity in an italicized, bold, or highlighted font.	**F**	CheckBookStockQuantity	**uc3**
17	20.4	If a search yields a result that has a stock quantity of zero, the system shall automatically check if that book is in an order.	**F**	CheckBookRestockDate	**uc4**
18	20.5	If that book is on-order, the soonest-expected restocking date shall be displayed …	**F**		
19	20.6	An option to browse the store's inventory by book's category, shall be available to all users.	**F**	BrowseByCategory	**uc5**
20	21.0	Searching existing orders shall only be explicitly performed by registered users	**F**		
21	21.1	An order search shall be done using one or multiple of the following categories: …	**F**	SearchOrderByID	**uc6**
22	21.2	Search results shall be displayed with perfect-matches first, followed by any other valid results …	**F**		
23	21.3	Once delivered, an order shall be added to the inventory in the system by a registered user.	**F**		
24	21.4	By default, a registered user shall be able to add an entire order to the inventory.	**F**	AddWholeOrderToInventory	**uc7**

ID	Para	Sentence	Category	Use Case Titles	UC ID
25	21.6	If that book was already on the inventory listing, the system shall update the inventory stock quantity by the amount on the order.	**F**		
26	21.7	If that book is not in the inventory, the book shall be added to the inventory using the book data from the order.	**F**		
27	21.8	In the event that not every book that was ordered was received …	**F**	AddPartialOrderToInventory	**uc8**
28	21.9	Adding a book to the inventory, without it being in an order, shall be a feature only available to administrators.	**F**	AddBookToInventory	**uc9**
29	21.10	Modifying a book's details in the inventory, shall also only be performed by an administrator.	**F**	ModifyDetailsForBook ChangeBookStockQuantity	**uc10** **uc11**
30	21.11	Likewise, removing a book entirely from the inventory shall only be done by administrators.	**F**	DeleteBookFromInventory	**uc12**
31	22.0	The system shall begin with one root administrator user.	**F**		
32	22.1	This user shall be able to create other administrator users or staff users.	**F**	CreateAdministratorAccount	**uc13**
33	22.2	Any administrator user shall be able to create or delete staff users.	**F**	CreateStaffAccount, DeleteStaffAccount	**uc14** **uc15**
34	22.3	Administrator user accounts shall only be deleted by the root administrator	**F**	DeleteAdministratorAccount	**uc16**
35	22.4	The root administrator account cannot be deleted.	**F**		
36	22.5	All registered users shall have a username and password used to log into the system.	**F**	LogIntoSystem	**uc17**
37	22.6	Resetting user passwords shall only be performed by administrators.	**F**	ResetPassword	**uc18**

ID	Para	Sentence	Category	Use Case Titles	UC ID
38	22.7	Once a registered user is logged in, the actions that particular user is allowed to perform shall be accessible.	**F**		
39	22.8	Actions that are exclusive to administrators shall not be accessible by staff users.	**F**		
40	22.9	All registered users shall be able to search existing orders	**F**	SearchOrderByOrderID, SearchOrderByISBN	**uc19** **uc20**
41	22.9	All registered users shall be able to … search the inventory	**F**		
42	22.9	All registered users shall be able to … add books to the inventory from delivered orders	**F**		
43	22.9	All registered users shall be able to … sell items to customers	**F**	ProcessSale	**uc21**
44	22.10	Administrator users shall be able to place new orders	**F**		
45	22.10	Administrator users shall be able to … create new registered user accounts	**F**		
46	22.10	Administrator users shall be able to … review sales information	**F**	GenerateProfitLossReport	**uc22**
47	22.10	Administrator users shall be able to … review system logs	**F**		
48	22.11	When an administrator creates a new user, the form to create the account shall have the following fields: …	**F**		
49	22.12	When an administrator deletes a user account, the action shall include a text field that gives a brief explanation of why the account is being deleted.	**F**		
50	23.0	Creating new orders shall be done by any administrator.	**F**	PlaceNewOrder	**uc23**
51	23.1	Orders … shall be populated by using the ordering interface to search for titles or ISBNs in the inventory or existing orders.	**F / IR**		

ID	Para	Sentence	Category	Use Case Titles	UC ID
52	23.2	If the desired book is in the search results, the interface shall allow that book to be added to the new order with a mouse-click.	**F / IR**		
53	23.3	If the book to be ordered has not been ordered previously, the administrator shall be able to create the entry with all the required fields filled in.	**F**		
54	23.4	Once an order is submitted, it shall receive a unique identifier that is associated with the order for future searches to reference	**F**		
55	23.4	Once an order is submitted, … it shall have a status of “in-progress”	**F**		
56	23.5	When an order is shipped and added to the inventory, its status shall be changed to “received”.	**F**		
57	24.0	Any registered user shall be able to sell a book.	**F**		
58	24.1	Once the user logs in, the ability to initiate a sale shall become available.	**F**		
59	24.2	Sales shall require a book’s ISBN.	**F**		
60	24.3	A book shall only be sold if it’s on the inventory list.	**F**		
61	24.4	The system shall be able to execute a sale for a book even if the stock quantity for that book is zero.	**F**		
62	24.5	When a book is sold, the in-stock quantity for that item on the inventory shall be reduced by one (or left at zero if the quantity is already zero).	**F**		
63	24.6	Refunding a book will only be performed by an administrator.	**F**	ProcessRefund	**uc24**
64	25.0	For every sale, the employee username, book details, quantity, and date shall be recorded for record-keeping.	**F**		

ID	Para	Sentence	Category	Use Case Titles	UC ID
65	25.1	An administrator shall be able to access sales records and query them by title, ISBN, author, keywords, genre, or sale date.	**F**		
66	25.2	The sale's features available to administrators shall allow the sales data to be queried using a range of consecutive dates …	**F**		
67	25.3	In this way, the user shall be able to view all sales over a given time period.	**F**		
68	25.4	Sales older than two years shall be archived outside of the reach of the system to prevent gradually slowing down the system …	**F**		
69	26	Administrator logs shall not be searchable at initial implementation.	**F**		
70	26.2	They shall be maintained as chronological entries where most recent entries are at the top.	**F**		
71	26.3	Each log entry shall contain the time-stamp of the event.	**F**		
72	26.7	Partial orders are added to inventory [shall be logged].	**F**		
73	26.7	Partial orders … the user log-in that accepted the order … shall be in the log entry	**F**		
74	26.7	Partial orders … the item(s) that were missing … shall be in the log entry	**F**		
75	26.8	Administrators create a new registered user [shall be logged].	**F**		
76	26.8	Administrators create … the username that was created shall be in the log entry.	**F**		
77	26.9	Administrators delete a registered user [shall be logged].	**F**		
78	26.9	Administrators delete … the username that was deleted … shall be in the log entry	**F**		

ID	Para	Sentence	Category	Use Case Titles	UC ID
79	26.9	Administrators delete … the explanation for why the user was deleted … shall be in the log entry	**F**		
80	27.0	Administrators shall have the ability to submit statements to the logs.	**F**	SubmitStatementToAdminLog	**uc25**
81	27.1	They shall not be able to edit the log files directly.	**F**		

4.4 NONFUNCTIONAL REQUIREMENTS

4.4.1 USABILITY

At a minimum, the anonymous user is expected to have an intuition about how to use a computer search. If presented with a long text-box with placeholder text that says "search by title" and a button that says "search," no other guidance should be necessary to search for books. Users that want to search by author, ISBN, or description will find those options near the search bar in the form of checkboxes, radio buttons, or a dropdown box, for example. Any user familiar with Google search, to include the "advanced search features" options, should be able to fully utilize the system without logging in.

Registered users will receive training as part of the hiring process. Staff users are expected to be hired with the same amount of technical expertise as a typical customer. Brief training on how to sell an item and accept orders will be required. An administrator account would be given to individuals in management positions. They would be expected to understand the basics of software like those in the Microsoft Office suite. Sales information for the system, if presented in a spreadsheet format, will be more intuitive for these individuals. Creating accounts and accessing logs should be straightforward since these will be forms and chronological listings, respectively.

The system will include documentation such as a manual that is structured in the form of how-to's for different functions. The staff at the book store are not required to have more than a high-school level of education. A highly technical manual would be less useful than one that used screenshots and lists to explain the process of an action from beginning to end. It should also explain how to handle possible complications that arise during the process.

4.4.2 RELIABILITY

The system will be installed on multiple computers. In the event that any one computer fails, business operations will not be impacted. Operations should only be hampered if all of the computers that perform sales were to be simultaneously down. It is expected that 7 to 10 computers will be using the software daily. If a terminal were to freeze or become unresponsive, restarting it in the middle of an action will not be damaging. In fact, it should be expected that staff would restart a terminal to troubleshoot a problem.

All information created by the system will be stored off-site, on a server reached via the internet. If the internet was to fail or the system was otherwise unable to communicate with the server, sales will still be possible. To accomplish this, each terminal will save a local copy of, at a minimum, books' ISBN and price listings as pulled from the server once every 24 hours. At the POS terminals, if the server is

unreachable, this listing would be used to complete the sale. Record of the sale will be saved locally and sent to the server when communication is restored.

All system exceptions will be logged in system log files viewable by system maintainers (not in the dedicated Administrator log files for the store). Exceptions, to the highest extent possible, should be handled. If files are not found, data cannot be saved, connections fail, etc., the user encountering the exception will be notified of the problem (especially if he/she is likely causing the error with their input or actions) and if the exception can be handled by referencing local copies of information or trying again after a pause, those options will be leveraged. Handling exceptions is a case-by-case process, so sweeping generalizations are difficult to make, but recognizing them in testing and handling them without burdening the user to solve the problem is preferred when possible. The system will not encounter an exception, silently dismiss it, and leave the user assuming their actions were completed when in actuality, the action failed.

4.4.3 PERFORMANCE

Performance of the system will hinge mostly on the local processes. The only internet-reliant part of any process is accessing a database on a server. Processing the contents of the files will all need to be done on the user's terminal. The time for processing the file contents, once they are retrieved, will depend on the way the information is stored and the implementation of the search in the source code. Since all searching will be done in parallel threads, it should feel nearly instantaneous to the user. If the user cannot get results for a search in less than 10 seconds, it would be considered very detrimental to the user's experience. Well-known book vendors, like Amazon or Barnes and Noble, will supply search results in less than 5 seconds.

4.4.4 SUPPORTABILITY

The Bookstore's long-term goal is to open an e-commerce store. Since the system already depends on a server-side database, the gradual changes to bring the interface features to a browser will be manageable. Ultimately, all information about the inventory and sales for the business will be seamlessly merged into a shared server allowing the client to modernize their store, reach a vastly larger market, and leave ample room for growth.

For the system that is delivered in November and described in this requirements analysis document, Code Monsters can provide maintenance for an annual fee. Code Monsters will provide the source code and the compiled software, ready to install, to the client. In the future, if bugs or deficiencies are found by the client, Code Monsters will make the necessary updates to the software to rectify the problems. If features need to be added that are beyond the scope of the original project, the development team and client can again work together to develop a project plan and implement any desired changes.

4.4.5 IMPLEMENTATION

The system can be implemented on computers operating on either Windows 8 (or higher) or Mac OS X. The computers will have access to the internet. This allows clients to flexibly integrate the system with their existing infrastructure. The system software, once executed, can be left running for extended periods of time without a user. It intended to be started when the store opens and run continuously so it is immediately available for customers or staff to use throughout the day. Some circumstances, such as loss of internet access, will cause the system to store information on the local terminal. For this reason, at least 4GB of hard drive space will be available for the system to write to and read from.

During system testing, the testing team will work with the client to accurately simulate conditions at the store. The team will mimic the operating systems, the user traffic, and the various scenarios likely

to be encountered after implementation. Normal conditions and fringe conditions will be tested. Testing will include assessing interactions with unrelated software also running on the terminal workstations. The testers will use only hardware and software currently used by the client and/or expected to be used by the client in the near future.

If the software encounters runtime errors, information about the exceptions will be written to maintenance logs which are viewable only by maintenance staff. The logs will not be viewable via the system interface regardless of user type. The maintenance team requires that the system software be installed by the root user of each terminal computer so that permissions can be enforced on log files preventing unauthorized access, modification, or deletion. To troubleshoot complications that may occur after launch, the root user will work with the maintenance team to access the logs and assess solutions.

4.4.6 INTERFACE

The interface should be very simple. The types of actions an anonymous user will need to perform are basic: search and read. The interface should have a minimum amount of complexity for searches. The search options will be ISBN, title, author, description/keywords, and genre. Once the search results are displayed, the interface will allow for rearrangement of results, such as ascending vs descending and ordering by title, author, price, or availability which will be shown on the results listing. The only input devices needed to run the system should be a keyboard and mouse.

Registered users should have a similarly simple interface and will rely on the same input devices. Since they will receive formal training at the time of hire, the number of actions they can take can be increased. As a registered user, they will have access to the same data but with more detail; the interface should accommodate this increase. Administrators will have the maximum amount of access and control of the system's features. They will be trained even more extensively and will be expected to fully understand the system. The interface they are presented with should change sufficiently to allow them to see any of the various reports, lists, and logs.

4.4.7 OPERATION

The administrator needs to be the one overseeing the system; he or she should be in charge of verifying that the data that's generated is accurate. The staff should be heavily involved in daily operations and maintenance, such as adding previously ordered items into the inventory and performing daily backups of the database.

4.4.8 PACKAGING

The customer will access the DropBox link provided via email and will download the file; they will run through the normal process of installing a file. The database will be setup by Code Monsters prior to the link being provided. Theoretically, the program can be installed on any number of computers. If issues arise throughout the installation process, the Code Monsters team can be reached via email. The main point of contact would be Dino Cajic: dinocajic@gmail.com.

4.4.9 LEGAL

The customer will be provided with the Enterprise Level license. Upon receipt of the program, the customer may install as many copies as they choose to, and maintain the program through a third-party company if they choose to do so. The code and the program will be the property of the customer.

Under no circumstance, unless required by law or presented clearly in writing, will any person or organization associated in the development of the Bookstore Software be liable for any damages, loss of data, corruption of data or any other loss, expense or damage. Code Monsters and the personnel

affiliated with the development, presentation or delivering of the software, will not be liable for any situations that may arise out of the use of, or inability to use the software, such as, but not limited to, claims, cases or actions involving infringement of copyright, patent, trademark or trade secrets.

Code Monsters does not warrant that the program will meet your requirements or that the program will be error-free.

4.5 DATABASE

The database to be used with this system is a relational database, specifically a MySQL Relational Database. A MySQL database was chosen for the following reasons:

- Extensive use of SQL as a querying language; SQL reduces development time.
- Because the data is indexed, it's significantly faster to access.
- The company is looking to handle a large amount of data eventually, making the MySQL database easily scalable.
- The company's next step is an online e-commerce store.

The remainder of this section deals with database schematics.

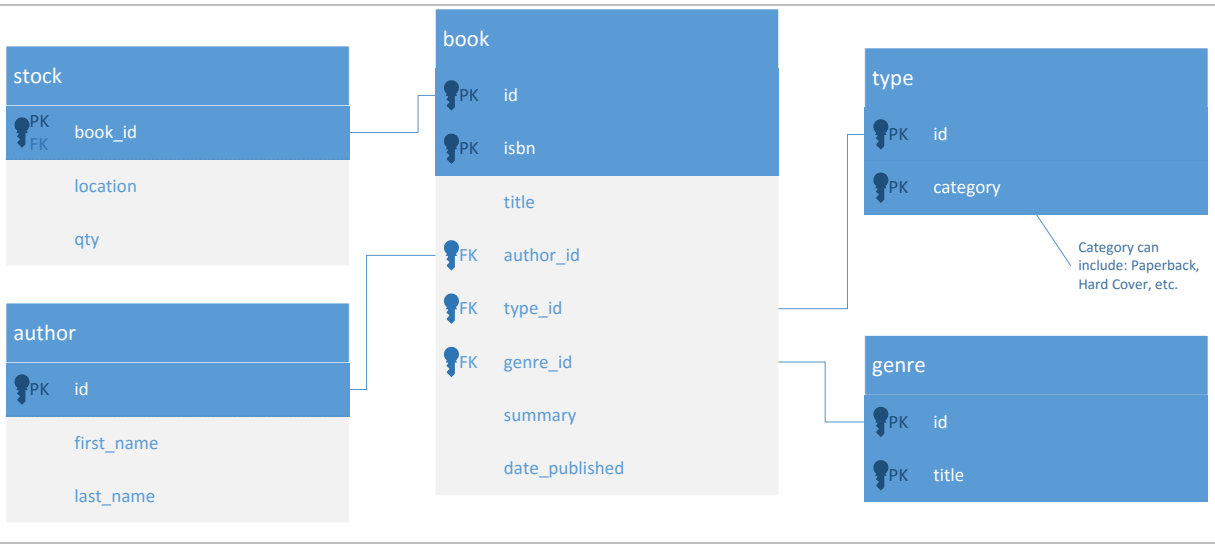

Figure 9: Tables involved in book

Each book is stored in the book table. The actual ISBN, summary of the book and the publication date are stored in the book table. Since ISBN's are unique, it has been added as a Primary Key. The author is stored in the author table. The author table contains the first and last name of the author. The id column in the author table is unique and set as the Primary Key. The book table and author table are linked via the author id. Each book can have a type. The types of books could be hard-back or paperback for now. The book type contains an id primary key and is linked to the book table by it. Each book also has a genre. The genre is an ever-expanding table with unique titles and id's. The genre table is linked to the book table via the genre id. A diagram of the table structure can be seen in Figure 9.

Each employee (shown in Figure 10) has to fill out an application with the HR department. Once the application has been approved, an administrator dealing with HR will log into the system and add the employee to the database. The employee's first name, last name, email, home address, phone number, emergency contact, date of employment and termination will be listed in the employee table. The employee type (separating administrators from staff) will be listed in the user_type table. The access

table stores the different privileges each employee may possess; the employee access table links the employee to the different possible privileges located in the access table.

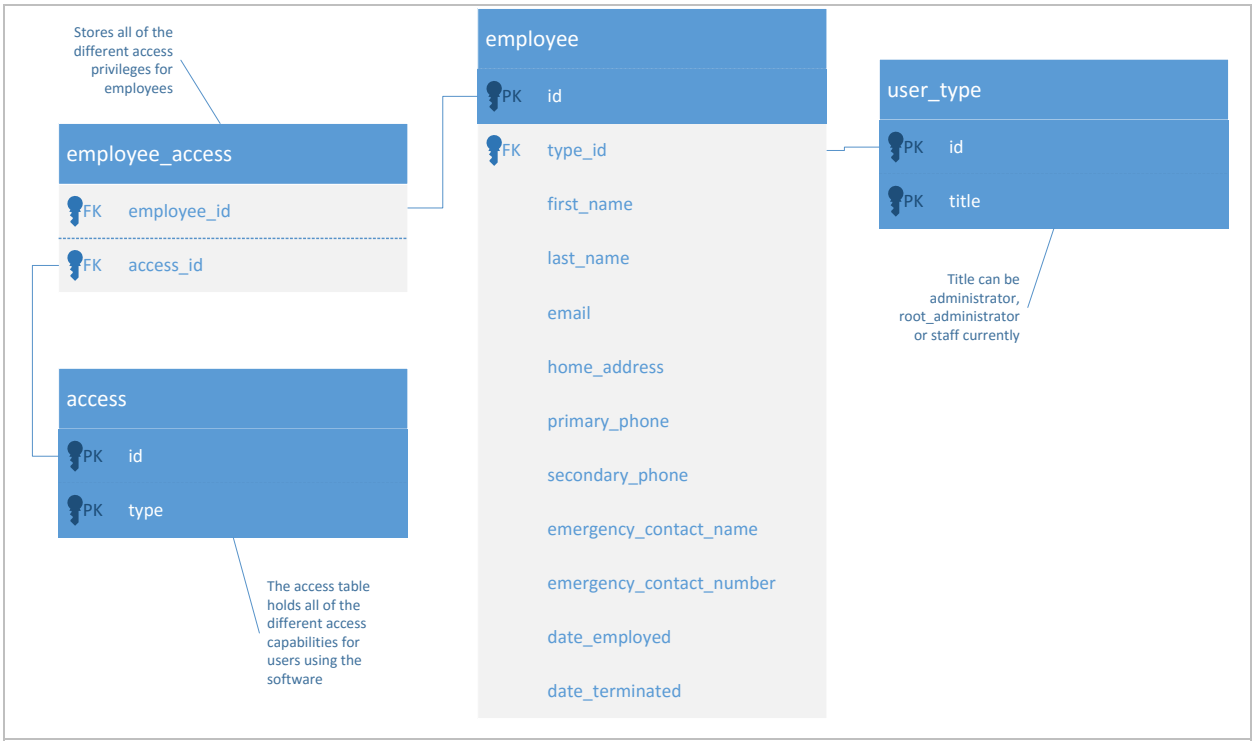

Figure 10: Tables involved in employee users

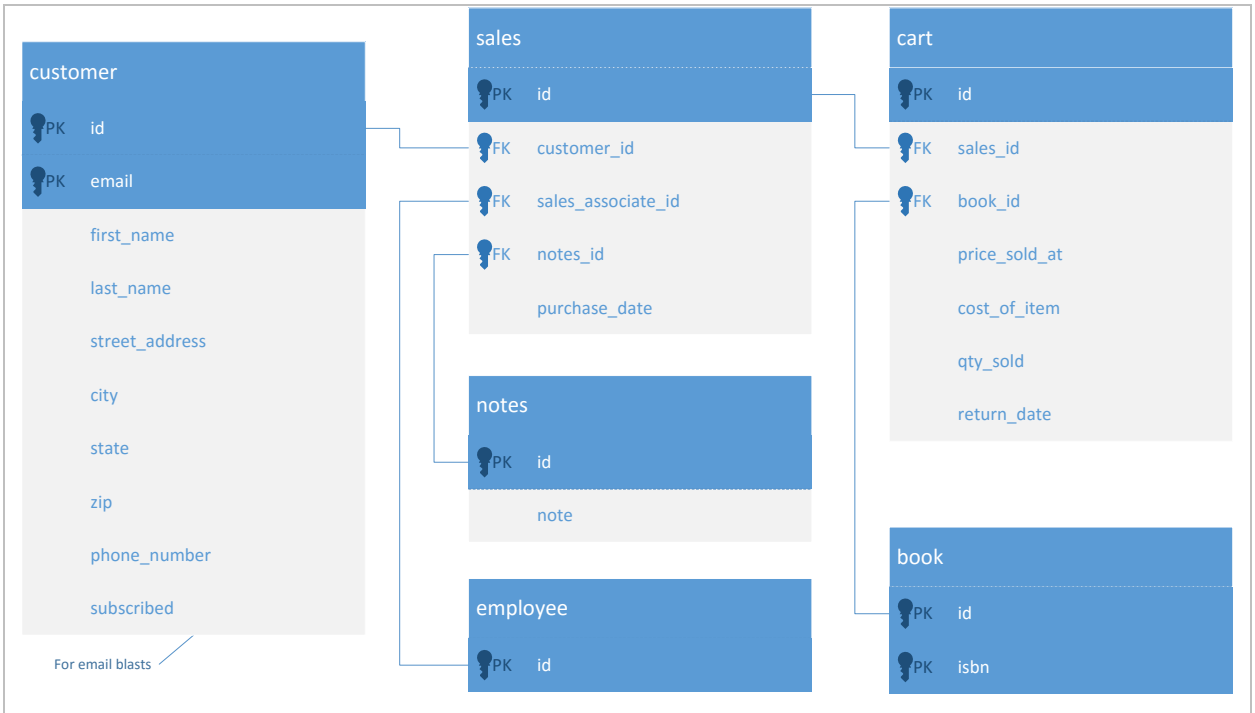

Figure 11: Tables involved in sales

Once a sale is processed, the information is stored in a few different tables. Customer information is stored in the customer table. This includes the customer's first and last name, the address, phone number and email address. Each item sold is stored in the cart table. The cart table contains the sales id, which helps link the items to the particular customer. It also contains the book id which links the cart to the book table (the book table contains all of the information concerning the book). The cart table also contains the price of the book at selling point, the cost of the item at selling point, and how many books were sold. The sales table helps link the different tables together, mainly the customer and cart tables. The sales table also contains the sales associate id that links the employee table to it. The sales associate is the person who processed the sale. Each employee is able to add special notes to the sale and the note is stored in the notes table. The sales table links to the notes table via the notes id. Lastly the sales table stores the purchase date. Since customers sometimes want to return individual items, and not necessarily all of the items, the return date has been added to the cart table. The diagram depicting the sales tables is shown in Figure 11.

4.6 FUNCTION POINT COST ANALYSIS

The following cost analysis was generated using COCOMO II – Constructive Cost Model. http://csse.usc.edu/tools/COCOMOII.php

Acquisition Phase Distribution

Phase	Effort (Person-months)	Schedule (Months)	Cost (Dollars)
Inception	1.8	2.3	$14,764
Elaboration	7.2	6.8	$59,054
Construction	22.7	11.3	$187,006
Transition	3.6	2.3	$29,527

Software Effort Distribution (Person-Months)

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	0.3	0.9	2.3	0.5
Environment/CM	0.2	0.6	1.1	0.2
Requirements	0.7	1.3	1.8	0.1
Design	0.3	2.6	3.6	0.1
Implementation	0.1	0.9	7.7	0.7
Assessment	0.1	0.7	5.4	0.9
Deployment	0.1	0.2	0.7	1.1

Total Customer Cost: $290,351

4.7 SYSTEM MODELS

The following section contains all of the Use Cases, Interaction Diagrams and Interface Diagrams for the proposed Bookstore Desktop Application.

4.7.1 USE CASES

Use case name	**AddWholeOrderToInventory**
Participating actors	Staff User or Administrator User
Flow of Events	1. The user confirms that every book on the order was received in the delivery. 2. The user finds the order by searching its order ID using the Order Search functionality. 3. When the order is found, the user can see the various options for that order, accessible by descriptive buttons. 4. The user clicks to the order's "Full Order Delivered" button. 5. The button changes colors and displays acknowledgement text once the operation has completed.
Entry Condition	• User is logged into the system at any terminal.
Exit Condition	• The order's status has changed from "in progress" to "received."
Quality requirements	• The time it takes to find an order should be less than 5 seconds. • The time it takes for an acknowledgement to be given after clicking "Full Order Delivered" should be less than 5 seconds.
Rationale for Use Case	Most orders will be received in-full. The option to accept an entire order with a single button click will be maximally efficient for the user.

Use case name	**AddPartialOrderToInventory**
Participating actors	Staff User or Administrator User
Flow of Events	1. The user has confirmed that a quantity of books is missing from the order that was delivered to the store. 2. The user finds the order by searching its order ID using the Order Search functionality. 3. When the order is found, the user can see the various options for that order, accessible by descriptive buttons. 4. The user clicks to the order's "Accept Partial Delivery" button. 5. A form for the order is displayed, pre-filled with the quantities of each book expected to be delivered. 6. The user changes the quantities for the books to reflect the actual quantity of books received and clicks the "Submit Order as Received" button. 7. The button changes colors and displays acknowledgement text once the operation has completed.
Entry Condition	• User is logged into the system at any terminal.
Exit Condition	• The order's status has changed from "in progress" to "received".
Quality requirements	• The time it takes to find an order should be less than 5 seconds. • The time it takes for an acknowledgement to be given after clicking "Submit Order as Received" should be less than 5 seconds.
Rationale for Use Case	The distributor will make mistakes. The client must have a way to update their inventory with received stock while also being able to account for inventory items they did not receive as expected.

Use case name	**AddBookToInventory**
Participating actors	Administrator User
Flow of Events	1. The user clicks on the “Inventory” tab. 2. The user clicks the “Add Book” tab and is prompted to input the book’s ISBN and press the “Submit” button. 3a. If the book is in the inventory, the form for the inventory item is displayed, pre-filled with that book’s current information. The in-stock quantity field is enabled. 4a. The user changes the quantity to the appropriate number. 5a. The user presses the “Update Book Quantity” button. **OR** 3b. If the book is not found in the inventory, an empty form for the book is displayed for the user to populate individually. 4b. The user enters all the appropriate information into the fields. 5b. The user presses the “Add New Book to Inventory” button. 6. The button changes color and displays acknowledgment text once the operation has completed.
Entry Condition	• User is logged into the system at any terminal.
Exit Condition	• The inventory reflects the change in quantity for the book.
Quality requirements	• The time it takes to display the appropriate form should be less than 5 seconds. • This use case can *include* the **ChangeBookStockQuantity** use case if the searched book is already in the inventory (steps 3a-5a). • The time it takes for an acknowledgement to be given after clicking the final button should be less than 5 seconds.
Rationale for Use Case	The client will find themselves with books to sell acquired through means other than ordering from the normal distributor. He needs a way to track those books from acquisition to sale in the system.

Use case name	**DeleteBookFromInventory**
Participating actors	Administrator User
Flow of Events	1. The user clicks on the “Inventory” tab. 2. The user clicks the “Remove Book” tab and is prompted to input the book’s ISBN and press the “Submit” button. 3. The book’s ISBN, Title, and Author are displayed with a message confirming that this is the book to be deleted. 4. If the user verifies this is the book they wish to delete, he or she presses the “Remove” button. 5. The button changes color and displays acknowledgment text once the operation has completed.
Entry Condition	• User is logged into the system at any terminal.
Exit Condition	• A search for the book shows the book is no longer in the system’s inventory.
Quality requirements	• The time it takes to display the confirmation message should be less than 5 seconds. • The time it takes for an acknowledgement to be given after clicking the “remove” button should be less than 5 seconds.
Rationale for Use Case	The client will need to remove books from their inventory over time if they choose to no longer carry that title.

Use case name	**PlaceNewOrder**
Participating actors	Administrator User
Flow of Events	1. The user clicks on the “Inventory” tab. 2. The user clicks the “Create New Order” tab. 3. Multiple inputs for ISBNs are displayed. 4. The user inputs the target books’ ISBNs and presses the “Submit” button. 5. For every ISBN that is not found in a current order or the inventory, a form is displayed to input all of that “new” book’s details. 6. After the details are inputted and the “Save Details” button is pressed for each of the books, the full order is displayed showing the books’ ISBN, Title, Author(s), Cost, and Price as a list. 7. Beside each book listed, an input box for “Quantity” is enabled. 8. The user inputs the quantity for each book they wish to order. 9. The user presses “Submit New Order” when all quantities are entered. 10. The button changes color and displays acknowledgment text once the operation has completed.
Entry Condition	• User is logged into the system at any terminal.
Exit Condition	• An order with a unique identifier is viewable in the system containing the details of the new order.
Quality requirements	• The time it takes to display each appropriate form should be less than 5 seconds. • Any book ISBN in an existing order or in the inventory should not be manually re-entered by the user. • The time it takes for an acknowledgement to be given after clicking the final button should be less than 5 seconds.
Rationale for Use Case	The client will require the ability to order new books to replenish inventory after selling to customers.

Use case name	**SearchOrderByISBN**
Participating actors	Administrator User
Flow of Events	1. The user clicks on the “Inventory” (Point of Sale) tab. 2. The user clicks on the “Order Search” tab. 3. A form with various inputs are displayed. 4. The user inputs the Order Number and ISBN in their respective fields (leaving the others blank) on the form and presses “Search”. 5. The results of the search will be displayed.
Entry Condition	• User is logged into the system at any terminal.
Exit Condition	• The user can traverse the results using the interface.
Quality requirements	• The time it takes to find the results should be less than 5 seconds. • If no results are found, there should be an indication of such. The display should not remain unchanged when the “Search” button is pressed.
Rationale for Use Case	The client needs administrators to have the ability to determine if particular books are already ordered or not. This will minimize over-ordering books, under-ordering books, and improve team communication between shifts with less pass-down.

Use case name	**SearchOrderByOrderID**
Participating actors	Administrator User
Flow of Events	1. The user clicks on the “Inventory” tab. 2. The user clicks on the “Order Search” tab. 3. A form with various inputs are displayed. 4. The user inputs the Order Number the corresponding field (leaving the others blank) on the form and presses “Search”. 5. The results of the search will be displayed. All contents of that order will be displayed if an order with that ID exists.
Entry Condition	• User is logged into the system at any terminal.
Exit Condition	• The user can traverse the results using the interface.
Quality requirements	• The time it takes to find the results should be less than 5 seconds. • If no results are found, there should be an indication of such. The display should not remain unchanged when the “Search” button is pressed.
Rationale for Use Case	The client needs administrators to have the ability to review current and past orders to improve supply management.

Use case name	**CreateAdministratorAccount**
Participating actors	Root Administrator User
Flow of Events	1. The user clicks on the “Administrator” tab. 2. The user clicks on the “Create Administrator Account” tab. 3. A form with all of the required fields is displayed. 4. The user enters all of the appropriate information into the fields. 5. The user presses the “Create”. 6. The button changes colors and displays acknowledgement text once the operation has completed.
Entry Condition	• User is logged into the system at any terminal.
Exit Condition	• The new Administrator Account is available to be used to log into the system.
Quality requirements	• The time it takes for an acknowledgement to be given after clicking “Create” should be less than 5 seconds.
Rationale for Use Case	New Administrator Accounts must be made as new managers are hired by the client.

Use case name	**CreateStaffAccount**
Participating actors	Administrator User
Flow of Events	1. The user clicks on the “Administrator” tab. 2. The user clicks on the “Create Staff Account” tab. 3. A form with all required fields is displayed. 4. The user enters all the appropriate information into the fields. 5. The user presses the “Create” button. 6. The button changes colors and displays acknowledgement text once the operation has completed.
Entry Condition	• User is logged into the system at any terminal.
Exit Condition	• The new Administrator Account is available to be used to log into the system.
Quality requirements	• The time it takes for an acknowledgement to be given after clicking “Create” should be less than 5 seconds.
Rationale for Use Case	New Staff Accounts must be made as new employees are hired by the client.

Use case name	**DeleteAdministratorAccount**
Participating actors	Root Administrator User
Flow of Events	1. The user clicks on the “Administrator” tab. 2. The user clicks on the “Remove Administrator Account” tab. 3. A form requesting the target administrator account’s username is displayed. 4. The user enters the username of the account he wishes to delete. 5. The user presses the “Remove” button. 6. The button changes colors and displays acknowledgement text once the operation has completed.
Entry Condition	• User is logged into the system at any terminal.
Exit Condition	• The targeted Administrator Account is no longer able to log into the system.
Quality requirements	• The time it takes for an acknowledgement to be given after clicking “Remove” should be less than 5 seconds.
Rationale for Use Case	When managers leave the employment of the client, the client must be able to remove their accounts from the system to minimize unauthorized access.

Use case name	**DeleteStaffAccount**
Participating actors	Administrator User
Flow of Events	1. The user clicks on the “Administrator” tab. 2. The user clicks on the “Remove Staff Account” tab. 3. A form requesting the target staff account’s username is displayed. 4. The user enters the username of the account he wishes to delete. 5. The user presses the “Remove” button. 6. The button changes colors and displays acknowledgement text once the operation has completed.
Entry Condition	• User is logged into the system at any terminal.
Exit Condition	• The targeted Staff Account is no longer able to log into the system.
Quality requirements	• The time it takes for an acknowledgement to be given after clicking “Remove” should be less than 5 seconds.
Rationale for Use Case	When employees leave the employment of the client, the client must be able to remove their accounts from the system to minimize unauthorized access.

Use case name	**SearchBookByTitle**
Participating actors	Any User
Flow of Events	1. The user types in the title they would like to find in the "Search" input field. 2. The user clicks on the "Search by Title" option (default). 3. The user clicks the "Search" button. 4. The results of the search are displayed. 5. All pages of the results are possible by scrolling the results and using the "Next" and "Previous" buttons.
Entry Condition	• User is accessing an available terminal in the store.
Exit Condition	• The user can traverse the results using the interface.
Quality requirements	• The time it takes to find the results should be less than 5 seconds. • Title should be the default search option selected. • The full list of results should be viewable using buttons on the interface to page forward and backwards.
Rationale for Use Case	One of the most common searches customers and registered users will use will be the searching for a book by its title. This feature is key for the inventory system.

Use case name	**BrowseByCategory**
Participating actors	Any User
Flow of Events	1. The user clicks on the category they would like to view. 2. The system provides the books in order of newest to oldest, however the user may sort by title, price, etc. 3. All pages of the results are possible by scrolling the results and using the "Next" and "Previous" buttons.
Entry Condition	• User is accessing an available terminal in the store.
Exit Condition	• The user can traverse the results using the interface.
Quality requirements	• The time it takes to find the results should be less than 5 seconds. • The full list of results should be viewable using buttons on the interface to page forward and backwards.
Rationale for Use Case	Many customers will want a book of the type they prefer, but they will not necessarily have a title in mind. Browsing is a way to enable them to easily sift through available books without traversing the store.

Use case name	**CheckBookStockQuantity**
Participating actors	Any User
Flow of Events	1. The user types in the ISBN or book Title they would like to view. 2. The user clicks on the corresponding "Search By" option: "ISBN" or "Title". 3. The user clicks the "Search" button. 4. The results of the search are displayed. 5a. The user finds the book they are searching for on the display and observes the "Quantity In-Stock" value for that book. 5b. *Optional:* The user may click on the book to bring up further details. The in-stock value is displayed on the details page as well.
Entry Condition	• User is accessing an available terminal in the store.
Exit Condition	• The user can see the "Quantity In-Stock" amount for the book they are searching.
Quality requirements	• The time it takes to find the results should be less than 5 seconds. • If no book was found, the user should be notified of this. • If the quantity is zero, the restock date should be shown also.
Rationale for Use Case	Customers and store personnel will require the ability to verify a particular book is in-stock or not.

Use case name	**CheckBookRestockDate**
Participating actors	Any User
Flow of Events	1. The user types in the ISBN or book Title they would like to view. 2. The user clicks on the corresponding "Search By" option: "ISBN" or "Title." 3. The user clicks the "Search" button. 4. The results of the search are displayed. 5. The user finds the book they are searching for on the display and observes the expected date new copies of that book are to be delivered.
Entry Condition	• User is accessing an available terminal in the store and is searching for a book that is currently out-of-stock.
Exit Condition	• The user can see the expected restock date for the book.
Quality requirements	• The time it takes to find the results should be less than 5 seconds. • This use case *extends* the **CheckAvailablity** use case. It will be initiated anytime the Quantity In-Stock for a search result is 0.
Rationale for Use Case	Customers and store personnel will require the ability to verify a particular book is on-order if the in-stock quantity is 0.

Use case name	**GenerateSalesReport**
Participating actors	Administrator User
Flow of Events	1. The user clicks on the “Reports” tab and then the “Sales Report” tab. 2. The user searches the system’s sales’ data to display the sales information they wish to be on the report. (i.e. the user performs a search for sales figures between the dates 1 March 2016 and 1 October 2016 to bring up all sales statistics for that time period.) 3. Once the desired sales information is displayed, the user presses the “Generate Report” button. 4. The report will be displayed in the system’s default viewer for the report’s file extension and can be printed, saved, and moved as needed.
Entry Condition	• User is logged into the system at any terminal.
Exit Condition	• The report is created and opened in a separate file viewing application ready to handle additional manipulation.
Quality requirements	• The time it takes to generate the report and open the document should be less than one minute.
Rationale for Use Case	The client will need to see sales’ information in a structured way, save it in a familiar format that does not depend on the system to run, and move/save as he or she deems necessary.

Use case name	**GenerateProfitLossReport**
Participating actors	Administrator
Flow of Events	1. The user clicks on the “Reports” tab. 2. The user clicks on the “Generate Profit/Loss Report” tab. 3. The system provides two input fields where the user can enter a date range. After entering the date range, the user clicks “Submit.” 4. The system generates and displays the report.
Entry Condition	• User is accessing an available terminal in the store.
Exit Condition	• The user can traverse the results using the interface.
Quality requirements	• The time it takes to find the results should be less than 5 seconds. • Charts should be utilized to visualize the data presented.
Rationale for Use Case	Administrators will need to generate Profit/Loss reports to make sure that the business is on track financially.

Use case name	**LogIntoSystem**
Participating actors	Staff User or Administrator User
Flow of Events	1. Under the “File” tab, the user clicks the “Log In” tab. 2. An input labeled “User Name” and “Password” are displayed. 3. The user enters their user name and password into the fields. 4. The user presses the “Log In” button. 4. The “Log In” tab changes to the user’s username and adds a “Log Out” tab underneath.
Entry Condition	• User is accessing an available terminal in the store.
Exit Condition	• The “Log In” button is now replaced with the user’s user name
Quality requirements	• The time it takes to display the “User Name” and “Password” fields should be less than 3 seconds. • The time it takes to log into the system after pressing “Log In” should be less than 5 seconds.
Rationale for Use Case	Registered user accounts control access to certain functions. Logging in allows the system to distinguish different access levels for different registered users.

Use case name	**LogOutOfSystem**
Participating actors	Staff User or Administrator User
Flow of Events	1. The user presses the “Log Out” tab under the “File” tab. 2. The “Log Out” tab is replaced with a “Log In” tab.
Entry Condition	• User is logged into the system at any terminal.
Exit Condition	• The “Log In” tab is now displayed where the user name was.
Quality requirements	• The time it takes to log out of the system after pressing “Log Out” should be less than 5 seconds.
Rationale for Use Case	The system must be able to switch users so the terminals can be shared by any user.

Use case name	**ResetPassword**
Participating actors	Administrator User
Flow of Events	1. The user clicks on the “Administrator” tab. 2. The user clicks on the “Reset a user’s password” tab. 3. An input box for the user name of the account to be reset is displayed. 4. The user enters the username and clicks Search. 5. The resulting username is displayed (if currently in the system) along with two new fields where the user can enter a password and retype it for verification purposes. 6. The user clicks submit to process the password change. 7. A confirmation message is displayed.
Entry Condition	• User is logged into the system at any terminal. • The username must exist in the system.
Exit Condition	• The reset password can be used by the specific user to log into the system.
Quality requirements	• The time it takes to display the input field for the user name should be less than 3 seconds.
	• The time it takes to display the confirmation method should be less than 5 seconds.
Rationale for Use Case	Registered users will inevitably forget their passwords and will require a way to reset them.

Use case name	**ChangeBookStockQuantity**
Participating actors	Administrator User
Flow of Events	1. The user clicks on the “Inventory” tab. 2. The user clicks the “Adjust A Book’s Stock Quantity” tab. 3. An input for the book’s ISBN is displayed. 4. The user inputs the target book’s ISBN and presses the “Submit” button. 5. The form for the inventory item is displayed, pre-filled with that book’s current information. The in-stock quantity field is enabled. 6. The user changes the quantity to the appropriate number. 7. The user presses the “Update Book Quantity” button. 8. The button changes color and displays acknowledgment text once the operation has completed.
Entry Condition	• User is logged into the system at any terminal.
Exit Condition	• The inventory reflects the change in quantity for the book.
Quality requirements	• The time it takes to display the appropriate form should be less than 5 seconds. • The time it takes for an acknowledgement to be given after clicking the final button should be less than 5 seconds.
Rationale for Use Case	The client will require that some books’ stock quantities be adjusted to reflect loss events other than sales and stock increases other than orders.

Use case name	**ModifyDetailsForBook**
Participating actors	Administrator User
Flow of Events	1. The user clicks on the “Inventory” tab. 2. The user clicks the “Adjust A Book’s Details” tab. 3. An input for the book’s ISBN is displayed. 4. The user inputs the target book’s ISBN and presses the “Submit” button. 5. The form for the inventory item is displayed, pre-filled with that book’s current information. All of the book’s fields are enabled. 6. The user makes changes as desired. 7. The user presses the “Update Book Details” button. 8. The button changes color and displays acknowledgment text once the operation has completed.
Entry Condition	• User is logged into the system at any terminal.
Exit Condition	• The inventory reflects the change(s) for the book.
Quality requirements	• The time it takes to display the appropriate form should be less than 5 seconds. • The time it takes for an acknowledgement to be given after clicking the final button should be less than 5 seconds.
Rationale for Use Case	The client will require the ability to change books’ details if mistakes are found, prices change, or other details need to be updated.

Use case name	**ProcessSale**
Participating actors	Staff User or Administrator User
Flow of Events	1. The user clicks on the “POS” tab. 2. The user clicks the “New Sale” tab. 3. A search box is displayed where the user can enter the ISBN for multiple books separated with a comma. 4. The user inputs the books’ ISBNs and presses “Submit” or hits the “Enter” key. 5. The books’ ISBN, Title, and Price are each displayed in a list with a subtotal, taxes, and total calculated. 6. If the customer pays with a Credit Card or Debit Card, the user presses the “Pay with card” button. If the customer pays with cash, the user inputs the amount paid into the “Cash Amount” input box and presses “Pay with cash”. 7. Once the transaction has been approved by the system (credit card verification), an acknowledgment will be displayed signifying the sale was successful.
Entry Condition	• User is logged into the system at any terminal.
Exit Condition	• The sale is recorded in the system’s sales statistics.
Quality requirements	• The time it takes to display the pricing page should be less than 30 seconds. • The time it takes for an acknowledgement to be given after clicking the final button should be less than 30 seconds.
Rationale for Use Case	The client requires the ability to process sales of books and track those sales in the same system.

Use case name	**ProcessRefund**
Participating actors	Administrator User
Flow of Events	1. The user clicks on the “POS” tab. 2. The user clicks the “Issue Refund” tab. 3. An input for the order is displayed. 4. The user input’s the book’s ISBN and presses the “Start Refund” button. 5. The book’s ISBN, Title, and Price is displayed with an input labeled “Alternate Refund Amount.” 6. The user input’s the refund amount if the price shown on the customer’s receipt is less than the book’s price. Otherwise, the input is left empty. 7. The user presses the “Issue Refund” button. 8. The button changes color and displays acknowledgment text once the operation has completed.
Entry Condition	• User is logged into the system at any terminal.
Exit Condition	• The user accesses the register and gives the customer their money back.
Quality requirements	• The time it takes to display the book’s details should be less than 5 seconds. • The time it takes for an acknowledgement to be given after clicking the final button should be less than 5 seconds.
Rationale for Use Case	The client will require the ability to issue refunds for reasons dictated by their company’s policies.

Use case name	**SubmitStatementToAdminLog**
Participating actors	Administrator User
Flow of Events	1. The user clicks on the “Administration” tab. 2. The user clicks on the “Create Log Entry” tab. 3. A form for log entry information is displayed for the user. 4. The user enters information into the appropriate fields as necessary. 5. The user presses the “Submit” button. 6. The button changes colors and displays acknowledgement text once the operation has completed.
Entry Condition	• User is logged into the system at any terminal.
Exit Condition	• The log shows the new information.
Quality requirements	• The time it takes for an acknowledgement to be given after clicking “Submit” should be less than 5 seconds.
Rationale for Use Case	Admin logs allow managers to communicate to each other via the system across shifts, have accountability, track important events, and monitor store activity. The ability to insert statements into the logs makes them flexible and unifies the communication system for management.

4.7.2 INTERACTION DIAGRAMS

Figure 12 depicts an Interaction Diagram for when a user searches for a book by title. The user will type the title into the search box and click search. The system will search for the book by title and retrieve the inventory for it. Multiple books matching that title will be loaded and displayed onto the screen. To help decrease latency, only 10 books will be loaded per page. A pagination system will be in place so that the user may load different titles if necessary.

After the user finds the book that he or she wants to read, the user will click on the book link to display further details about that particular book.

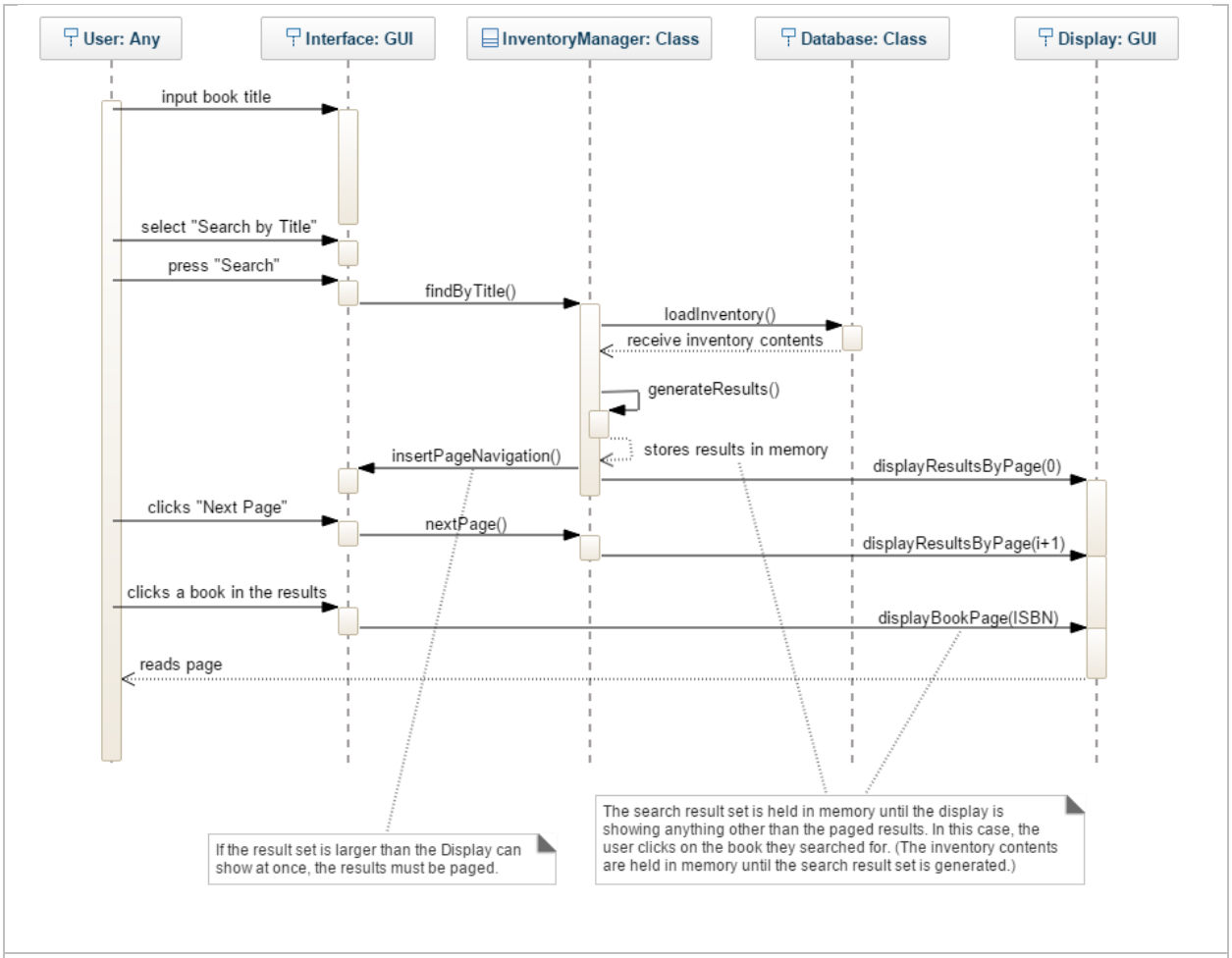

Figure 12: Search by Title Interaction Diagram

Any terminal running the software has the capability to allow the user to log into the system and unlock employee only features. The user will click on File and then Login. Once the user enters the username, password and clicks on the “Login” button, the system will log the user in by accessing the database and retrieving the credentials. The AccountMngr class will verify that the information submitted is correct and will update the session to state that the account has been successfully logged into. Figure 13 depicts this interaction between the user and the system.

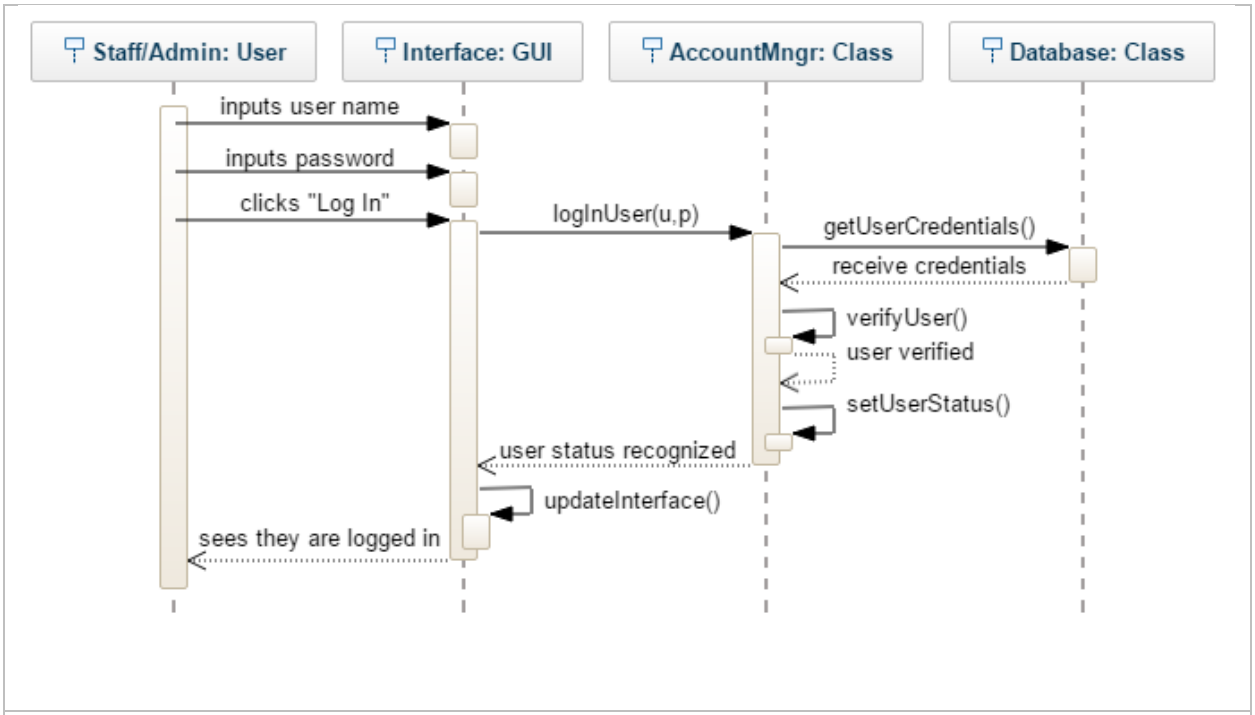

Figure 13: Log into the System Interaction Diagram

Once the user logs into the system, he or she will have access to certain function that were specified by the administrator when the employee's account was created. One such function will be the ability to process a sale. Figure 14 illustrates the interaction between the employee and the system when it comes to processing a sale.

The employee will have to be in the Point of Sale screen in order to process any order. The customer will provide a member of the staff with the book(s) that they're trying to purchase. The employee enters the ISBN and clicks next. The Sales class will load the content and display the current subtotal and items that are to be sold. An employee then notifies the customer of the subtotal and proceeds to enter the payment. Once the payment has been verified, the receipt is displayed and the inventory is updated to reflect the sold items. The customer is provided with a printout of the receipt and the merchandise purchased.

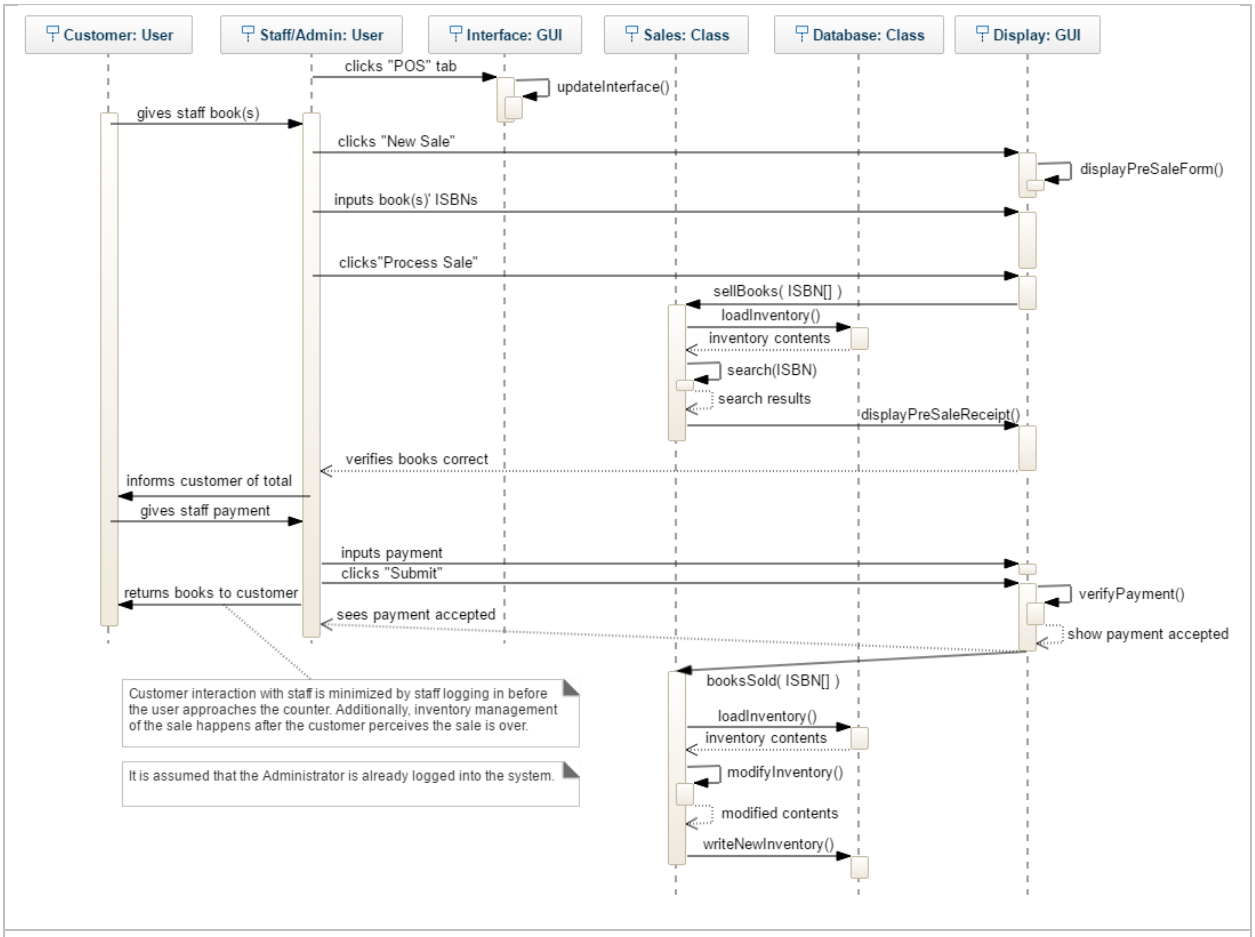

Figure 14: Process Sale Interaction Diagram

During any regular business transaction, a possibility of the customer returning the items purchased is always present. The system needs to be able to handle such instances and the Bookstore Desktop Application does just that.

The interaction diagram for returns processing is portrayed in Figure 15. The customer will provide the book that he or she wants to return to the employee. The employee will click on the Issue Refund tab located under the POS tab. A refund form is displayed allowing the employee to enter the book(s) ISBN(s). The employee clicks Process Refund to start the refund process.

The system will refund the book by doing the following:

- Searches to make sure that the ISBN is available.
- Displays a form to the employee to verify the refund amount.
- Once the employee clicks on the Verify Refund button, the refund is authorized and the display shows refund completed.
- The system updates the inventory by adding the part back into the system.

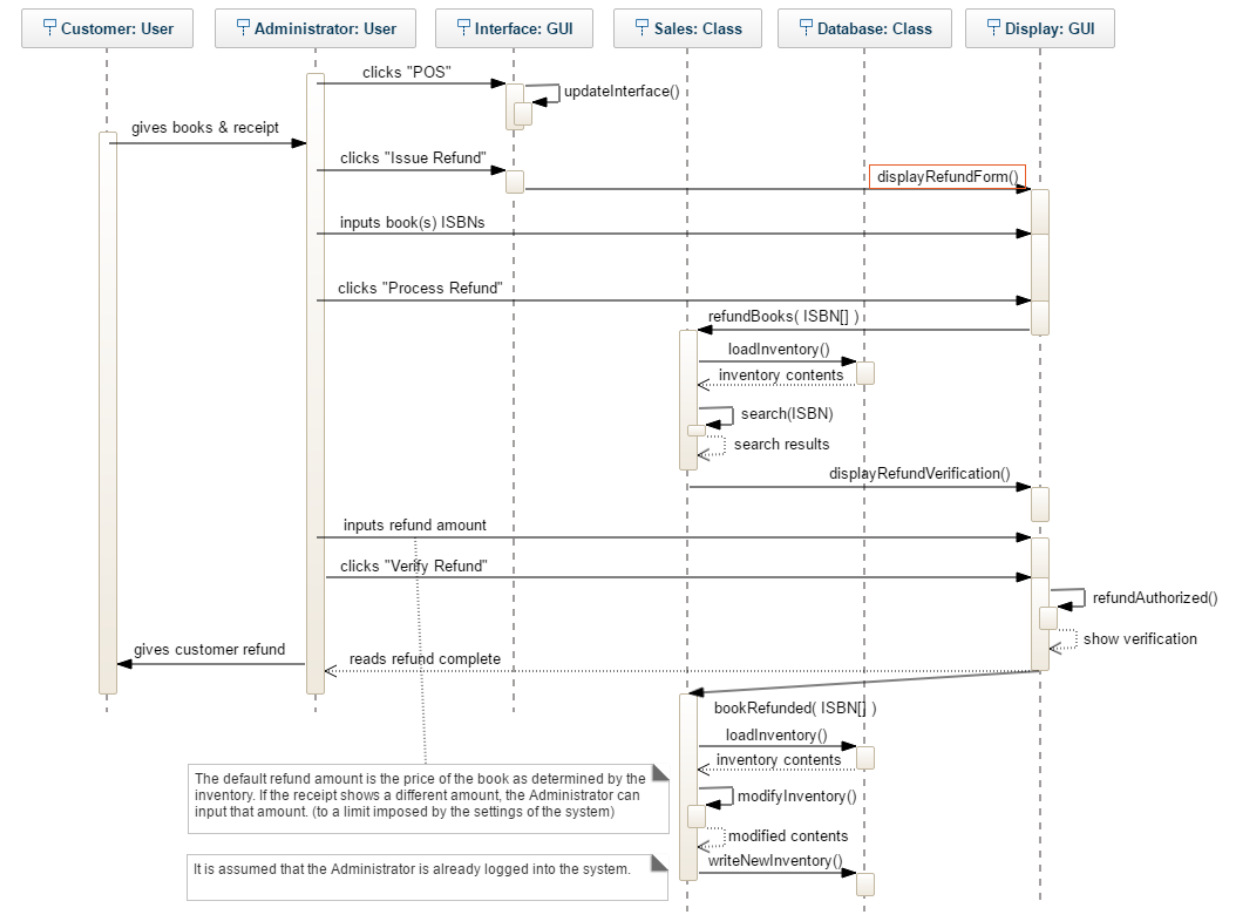

Figure 15: Process Refund Interaction Diagram

During any business's lifecycle, employee will have to be hired at various points in time. Administrators have the ability to add new employees to the system. Figure 16 represents an interaction diagram revolving around the notion of employees creating a user account.

The Administrative user clicks on the "Create Staff Account" tab located under the "Administrator" tab in the menu bar. A form is presented and provides the administrator input fields for entering employee data. Once the required information is entered, the user clicks the Submit button. The form is processed and the new employee is added into the database. A confirmation message is displayed to the employee letting him or her know that the user was successfully added to the system.

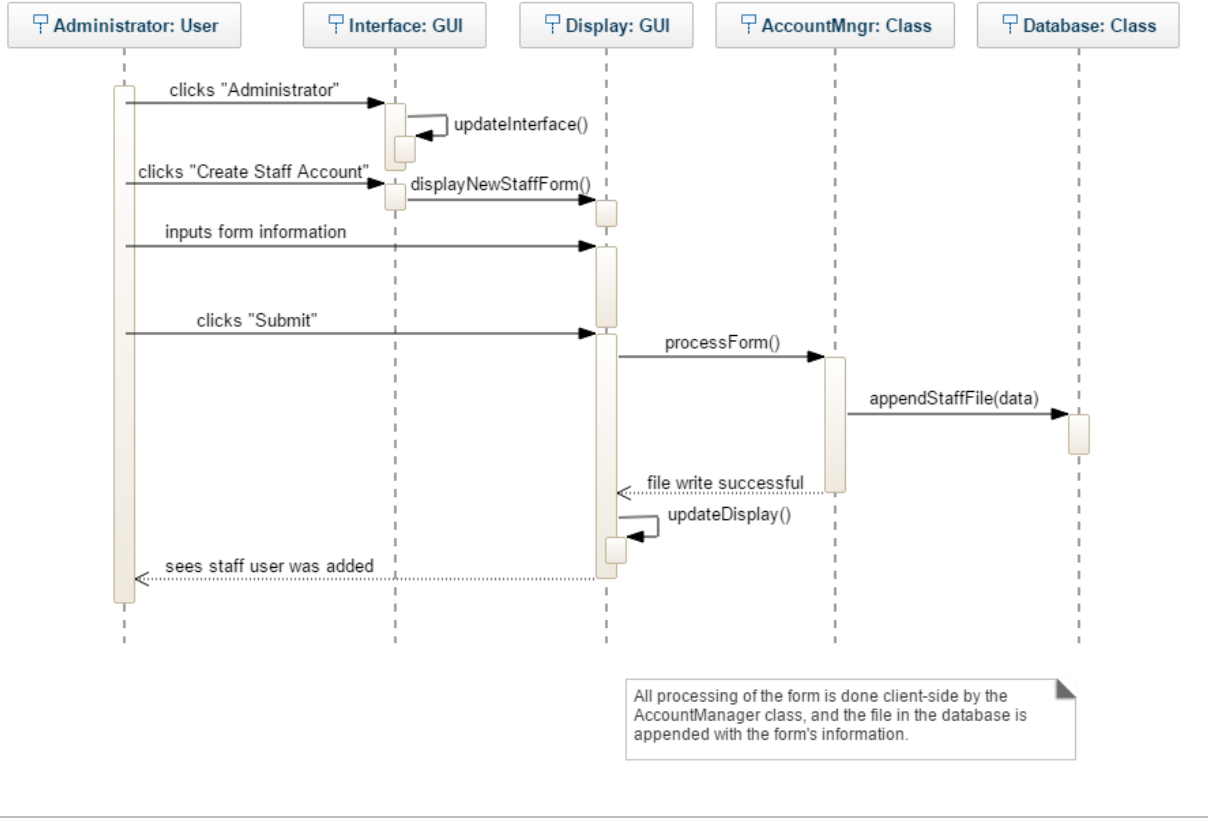

Figure 16: Create Staff Account Interaction Diagram

4.7.3 USER INTERFACE

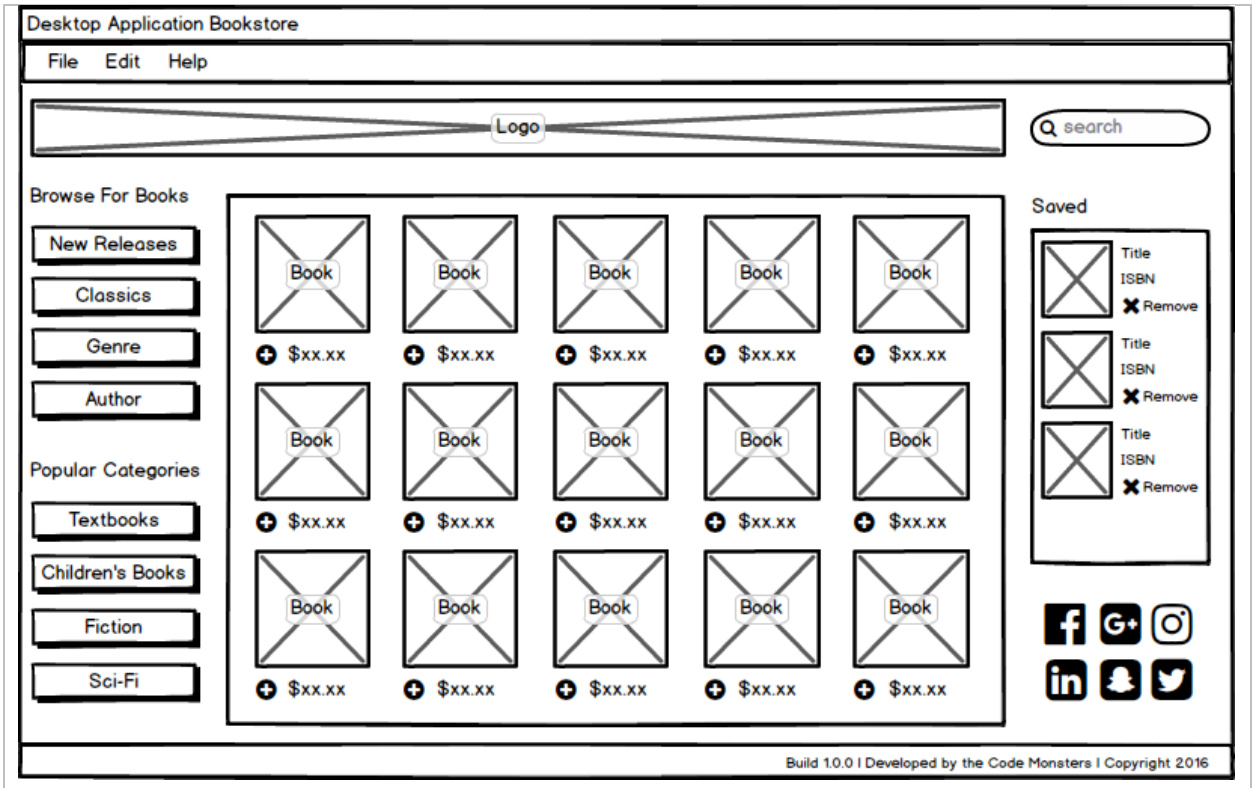

Figure 17: Customer View of Main Screen

Figure 17 represents what any user will see when the Bookstore Desktop Application is first launched. The top has simple navigation where the employees may access to log into the system. The Login tab is located under the File Tab. Immediately underneath the menu bar is the Company Logo and a Search Input to the right. Users may use the search to browse for books by title or ISBN.

The section underneath is divided into three columns. The far left column contains the Browse for Books by Category and a few other popular links that are to be decided on with the customer. The middle column lists all of the newest books that the bookstore has received. Clicking on the plus button will place the book into the far right column underneath the Saved section. The user may access these books at any point through their session.

Immediately underneath the Saved column are the Social Media links that the Bookstore is engaged in.

After the user has searched for a book, a results page will display 10 books on the page matching the title as closely as possible. The left portion of the screen allows the user to narrow down the search results by selecting the different types of available categories and the formats of the book. The user may also sort the results by cost, availability, title or year published. Each title is clickable allowing the user to access further details about the particular book. The results page is depicted in Figure 18.

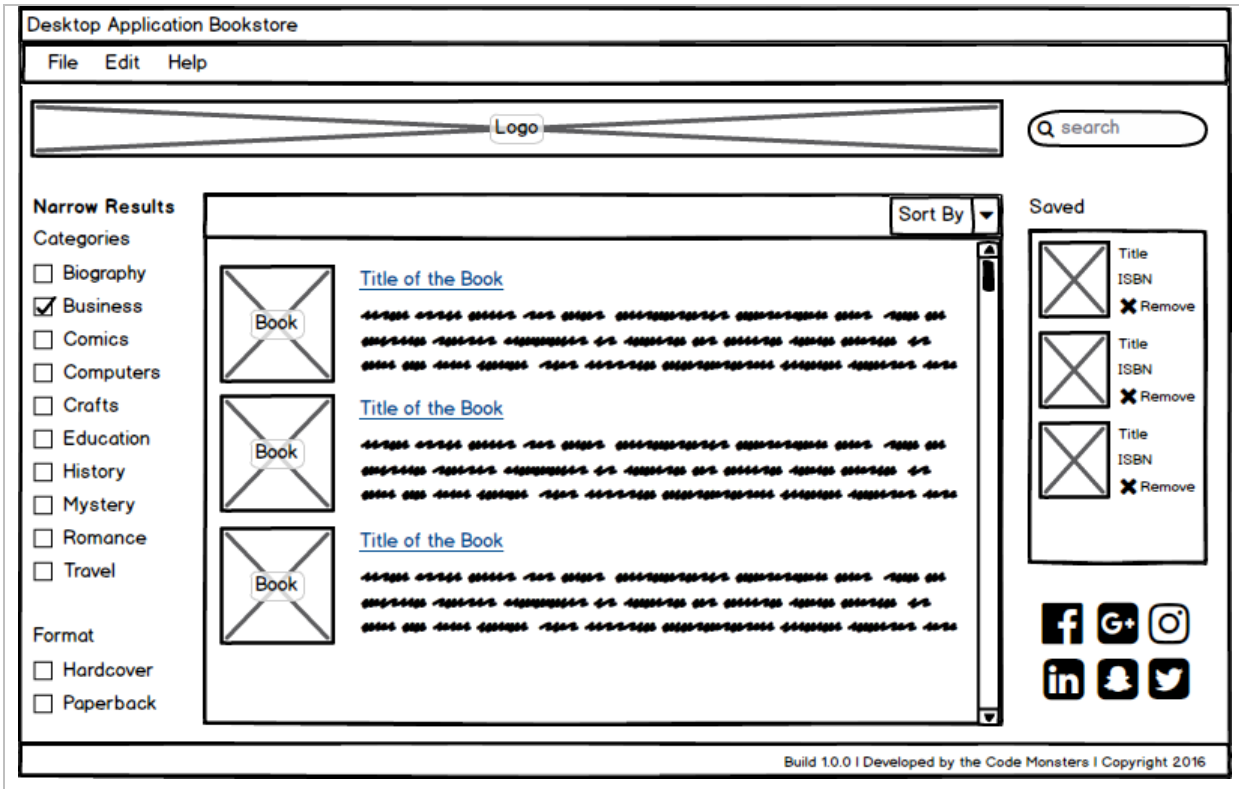

Figure 18: Customer View of Results Screen

Accessing the details of a particular book can be seen in Figure 19. The page contains the Title of the Book, the Author(s) that wrote the book and the publication date right next to the image. Immediately underneath the book's image is the plus button that allows a user to save the book for later viewing and the cost of the book.

Listed below is a detailed summary of the book. Whereas on the previous screen the user only saw a small summary, the details page sees the entire summary for that particular book. If the user chooses to eventually incorporate the e-commerce website and link it to the software, unbiased reviews will be listed below the summary.

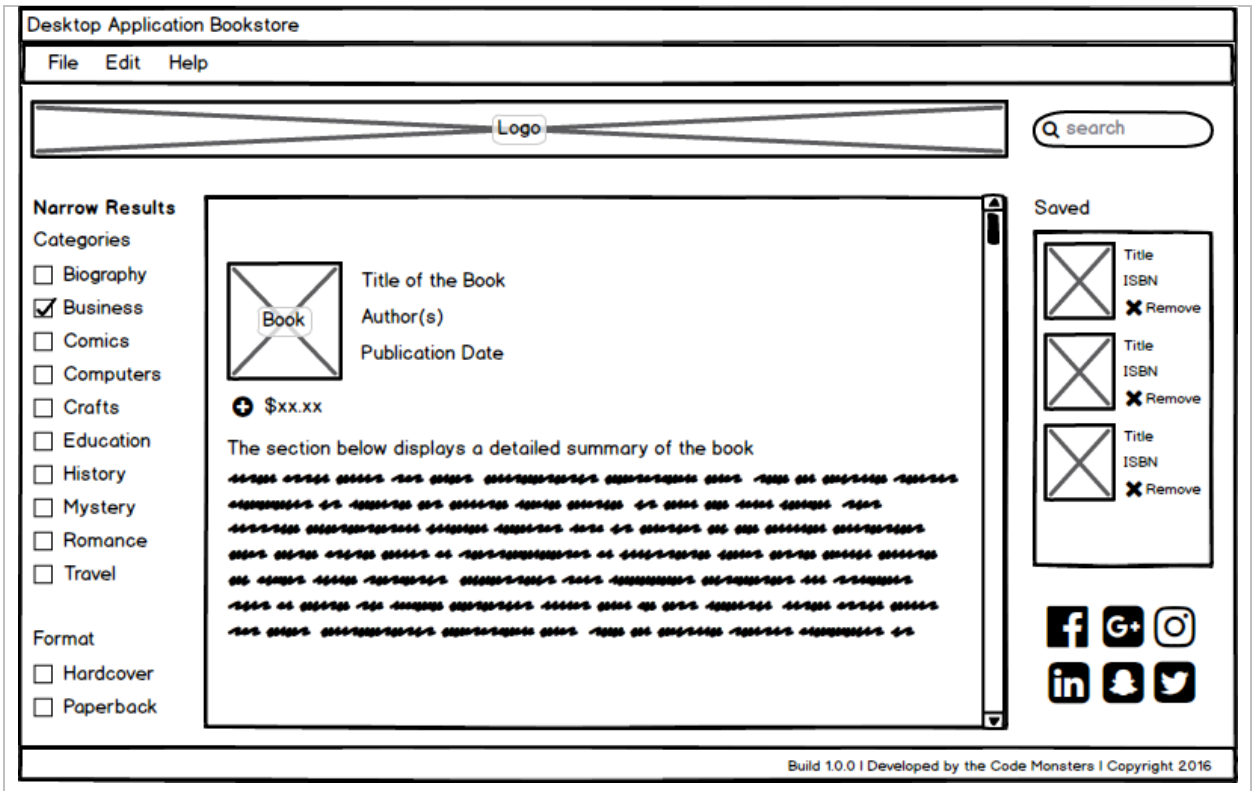

Figure 19: Customer View of the Book Details Screen

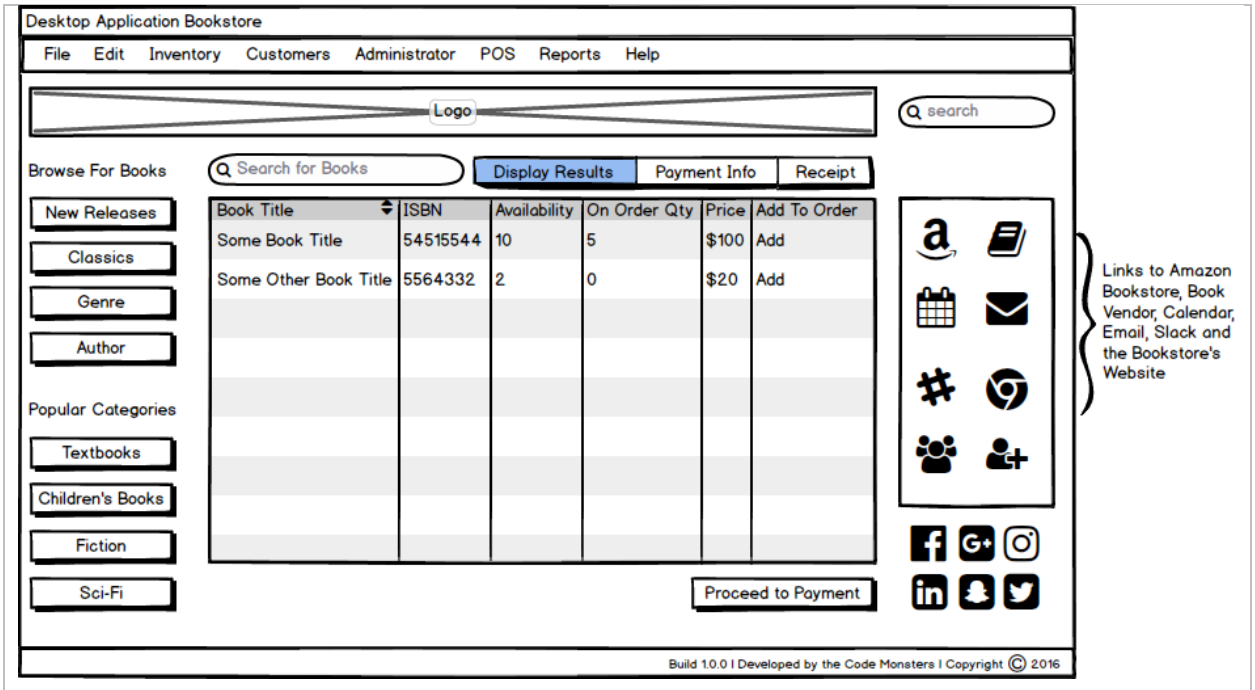

Figure 20: Employee View of POS Screen

Upon logging into the system, the employee will be shown a few additional options. The menu bar across the top will transform and display other key tabs that a member of the staff or an administrator might need to see. The right column has been transformed to include links that the employees will find useful. Currently, these links include:

- access to the Amazon webstore (since the client sells books on Amazon)
- a link to the book vendor to place new stocking order
- a link to the company calendar (currently the Google calendar web application)
- a link to Gmail
- a link to Slack for internal messaging
- the bookstore's current non e-commerce website

There are also a couple of quick links to add users and view the sales staff; these options are only visible to administrators.

Figure 20 shows the Point of Sale screen. To access the POS screen, the employee will click on the POS tab and select "New Sale" from the options provided. A progress bar will be displayed alongside the search input field. Once the user enters an ISBN or title of the book, he or she is able to add the items to order by clicking on the Add link.

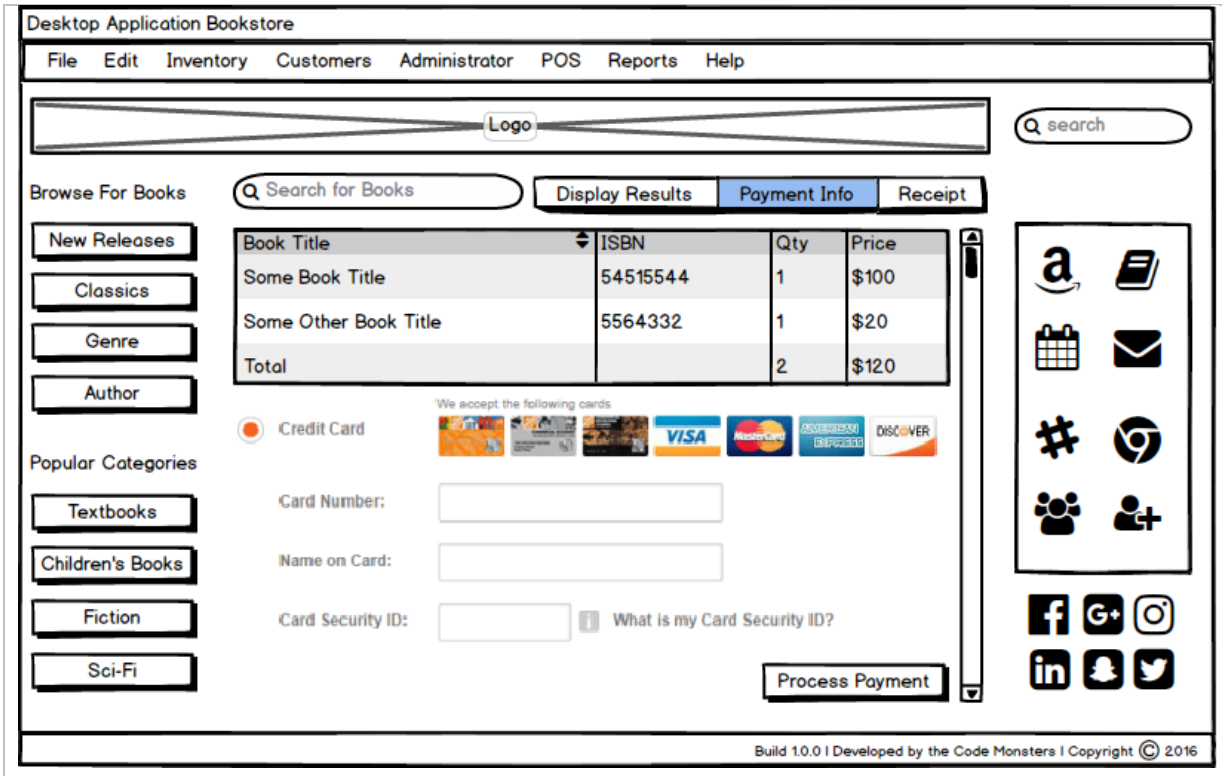

Figure 21: Payment Page

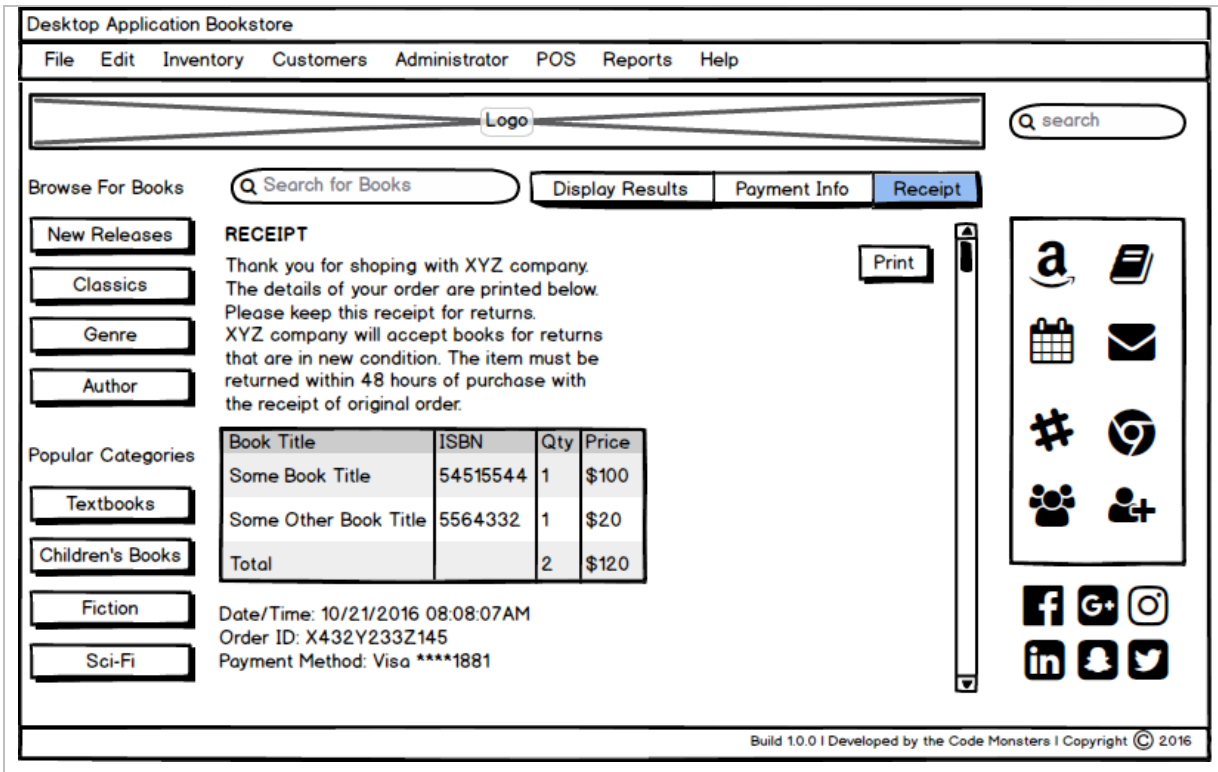

Figure 22: Receipt after payment completion

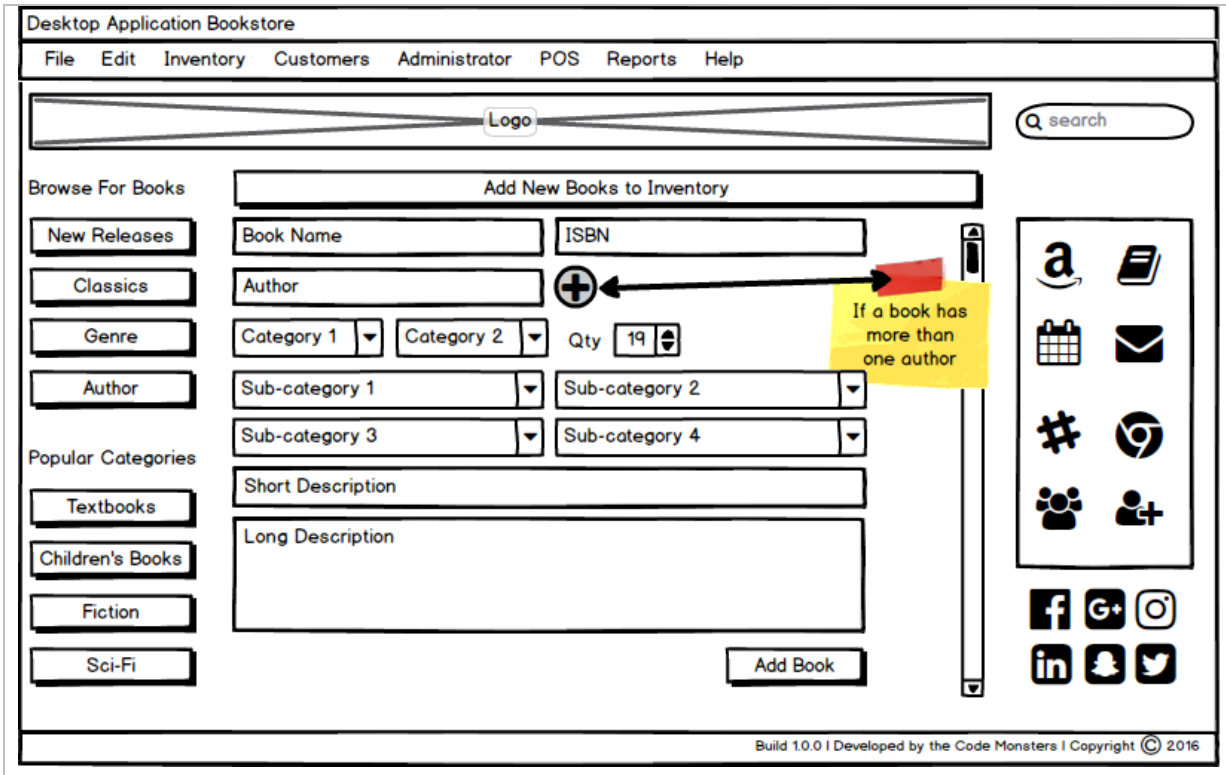

Figure 23: Add Book to Inventory

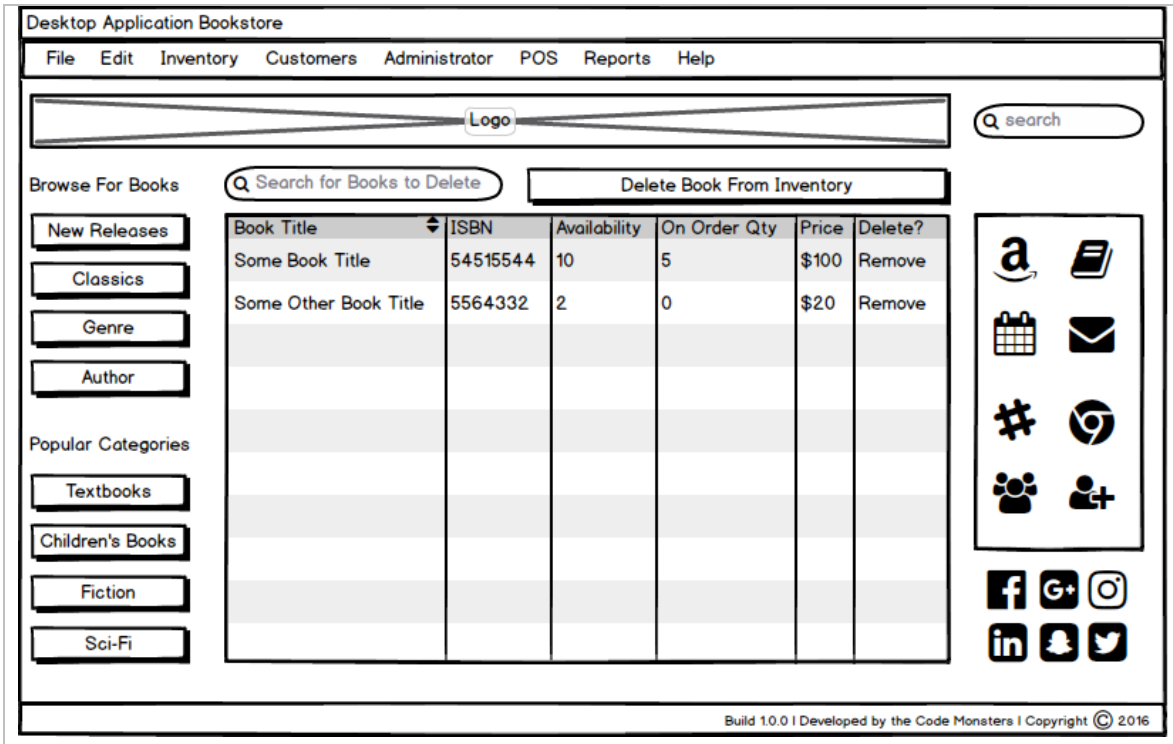

Figure 24: Delete Book from Inventory

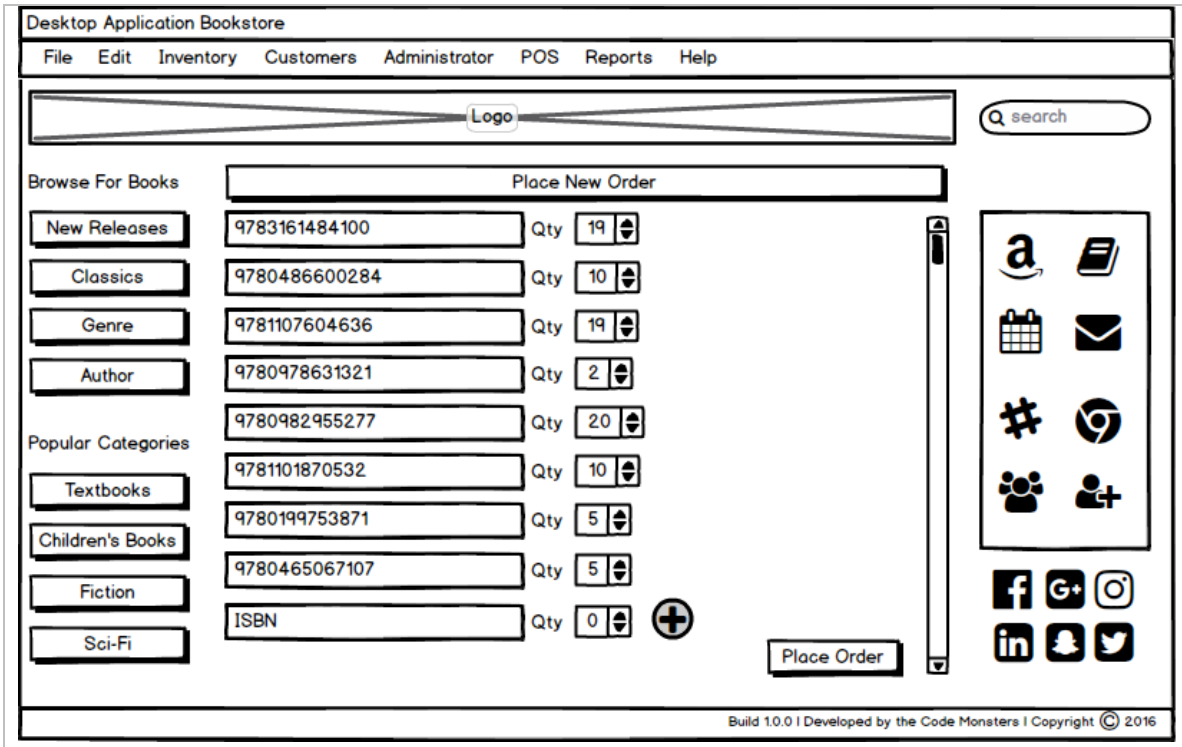

Figure 25: Place New Order

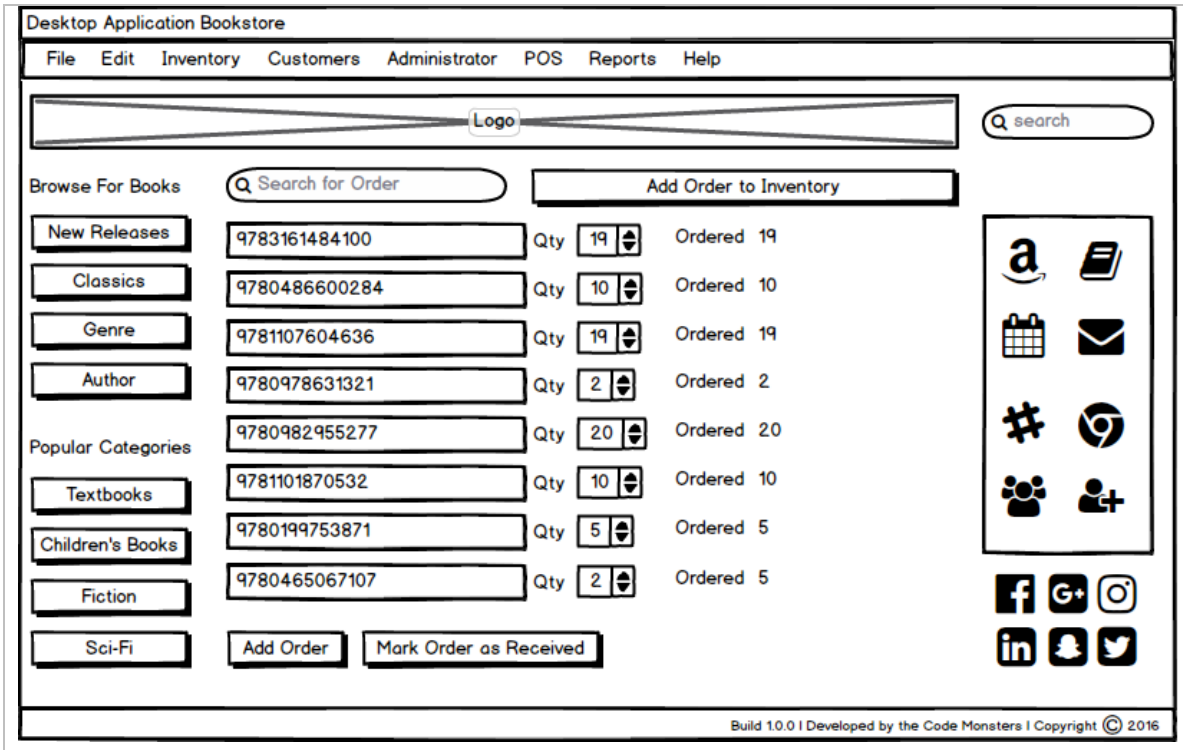

Figure 26: Add Order to Inventory

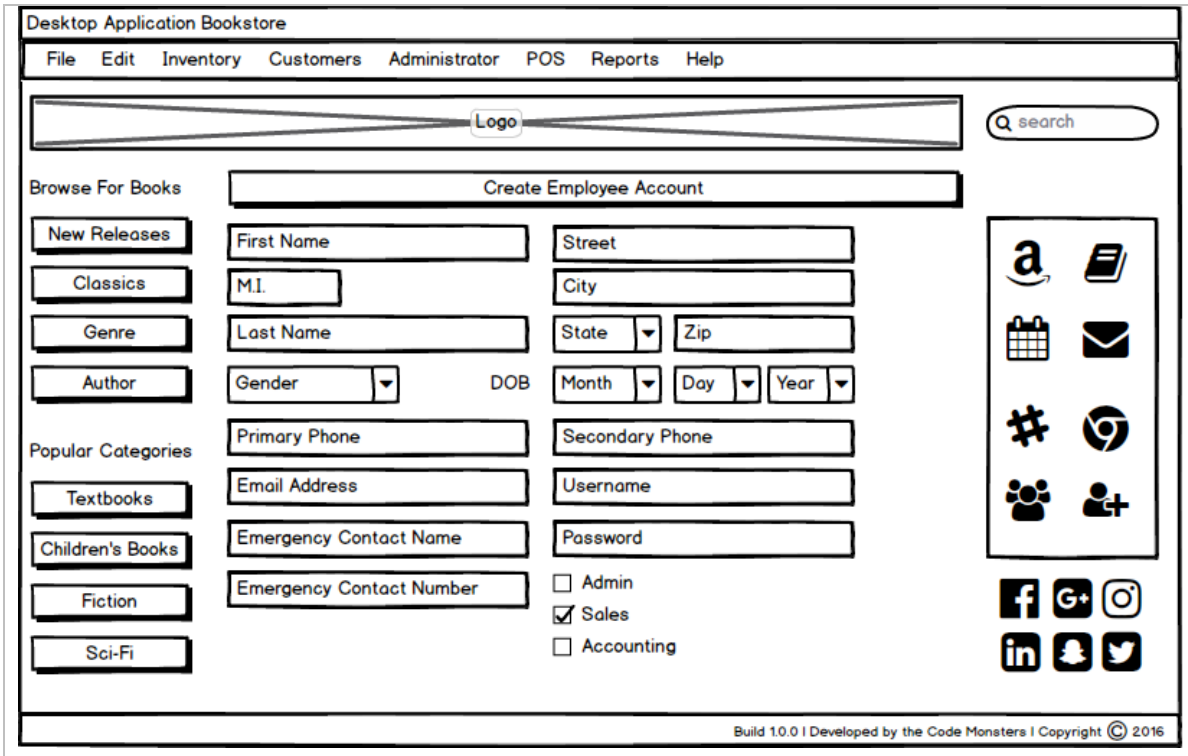

Figure 27: Create Employee Account

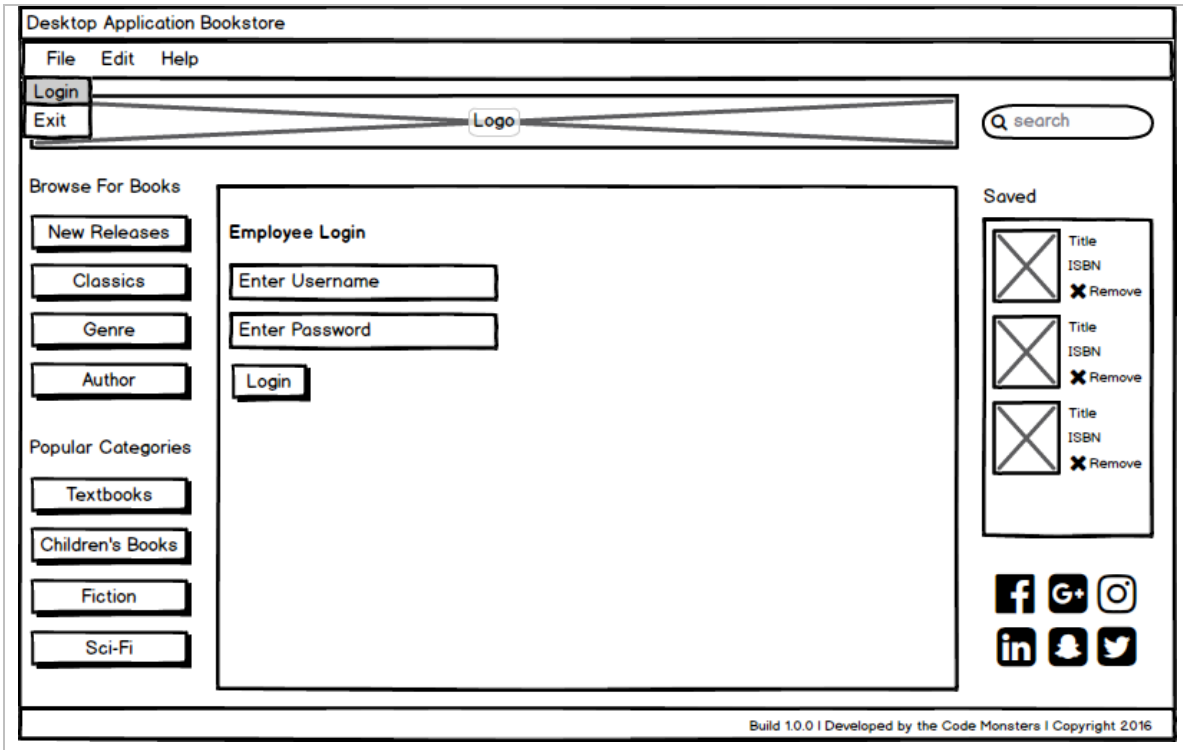

Figure 28: Employee Login

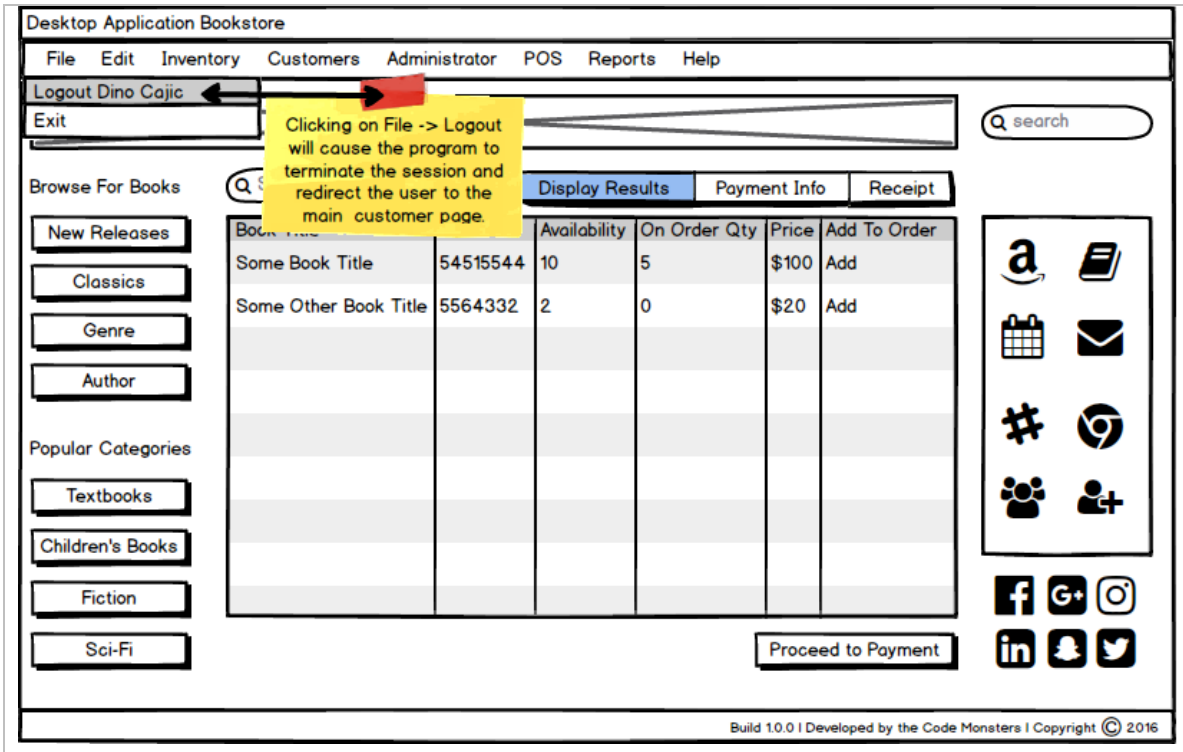

Figure 29: Employee Logout

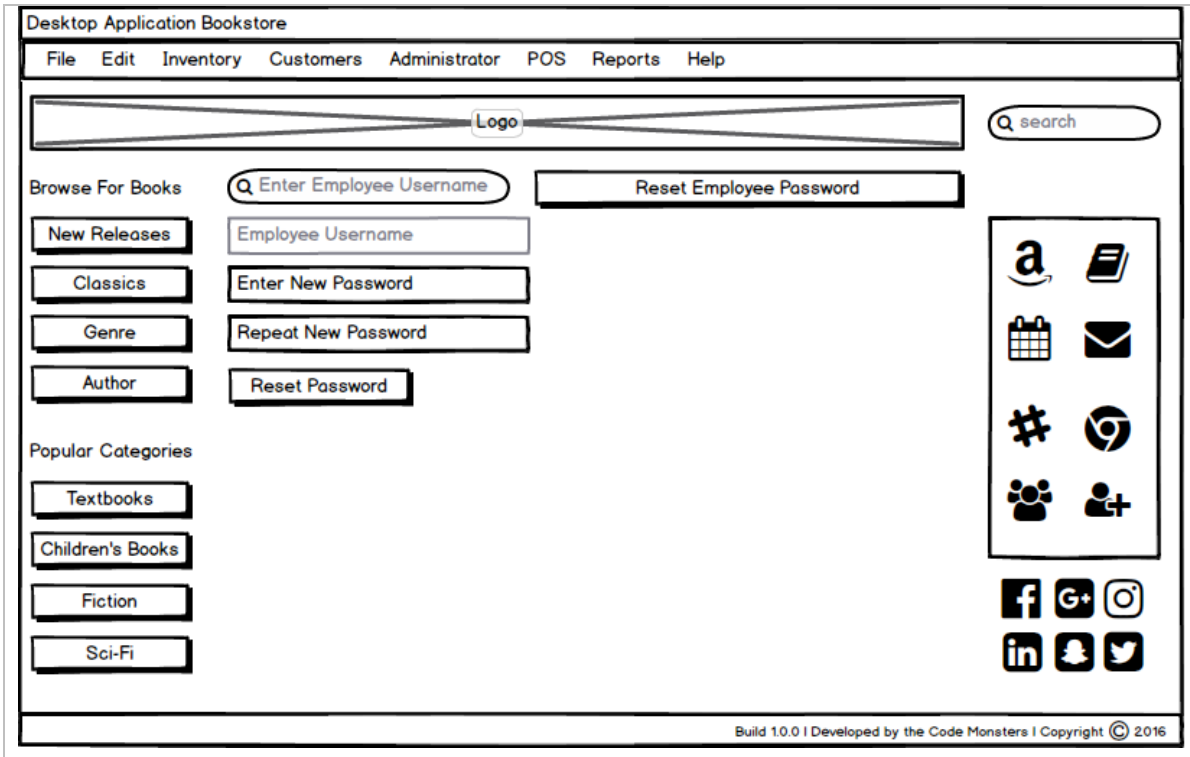

Figure 30: Resetting Employee Password

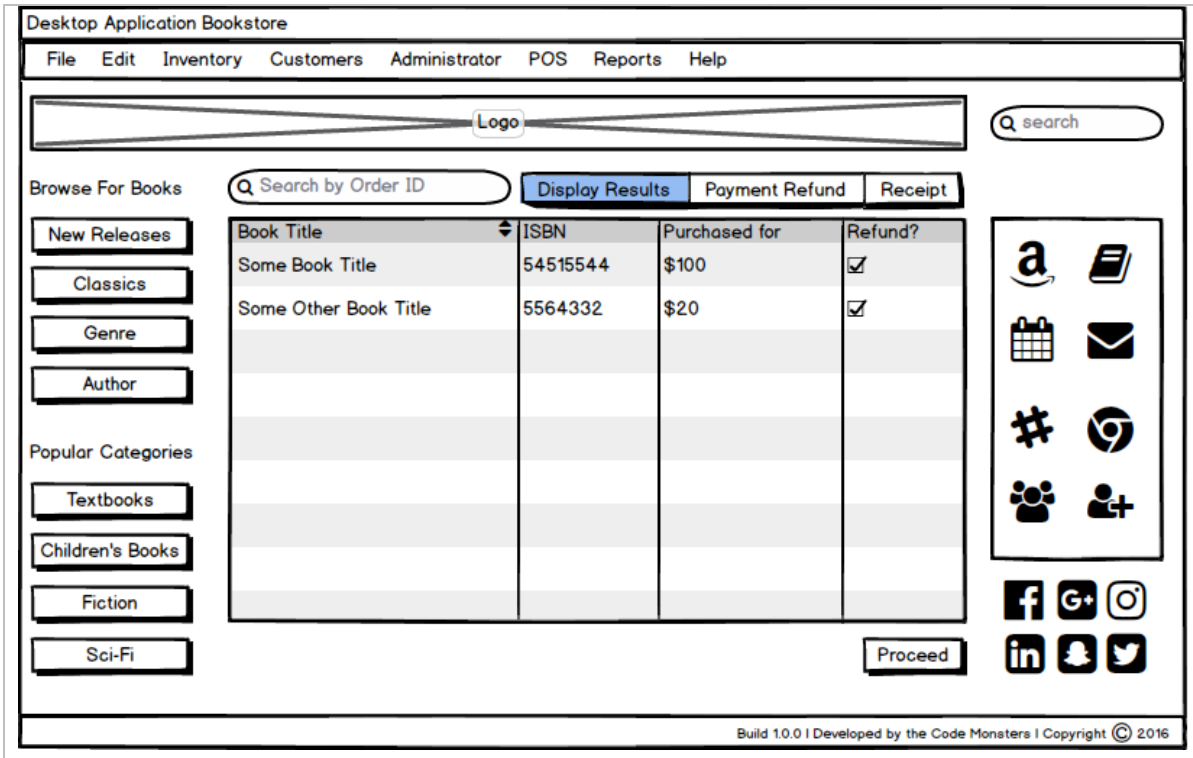

Figure 31: Processing Refund. Searching for the books to be refunded.

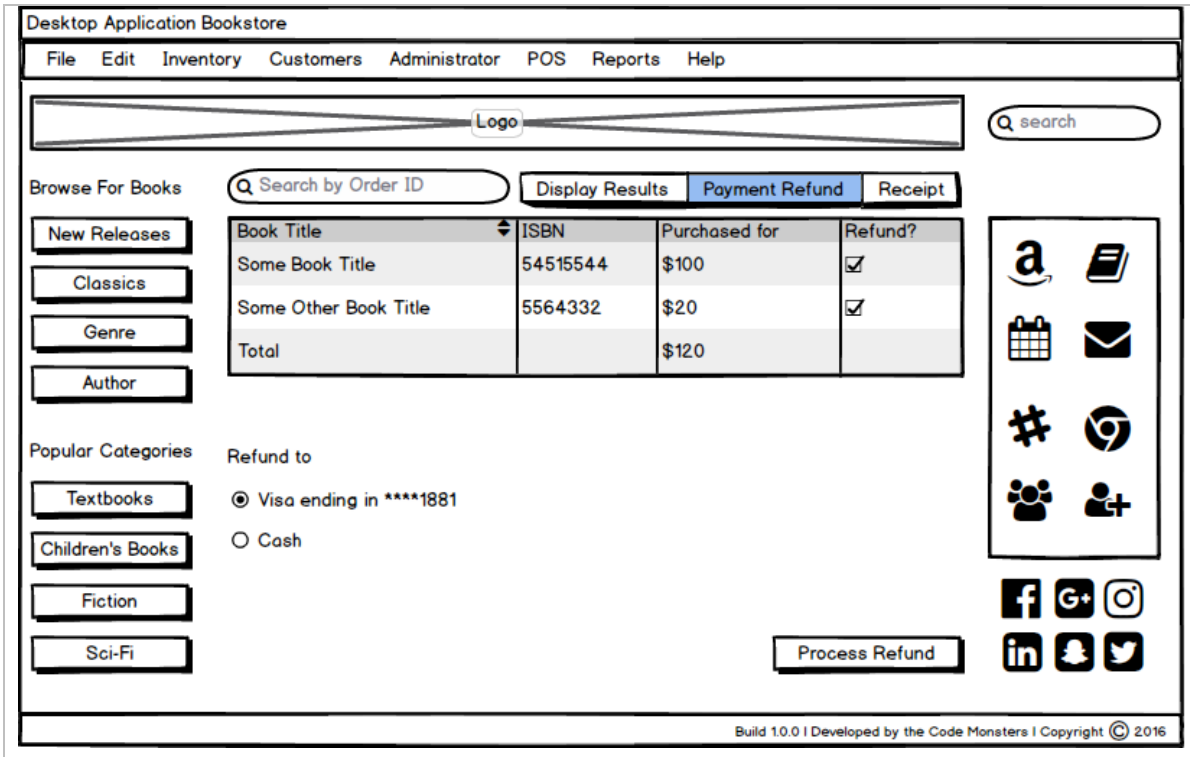

Figure 32: Processing Refund. Refunding to?

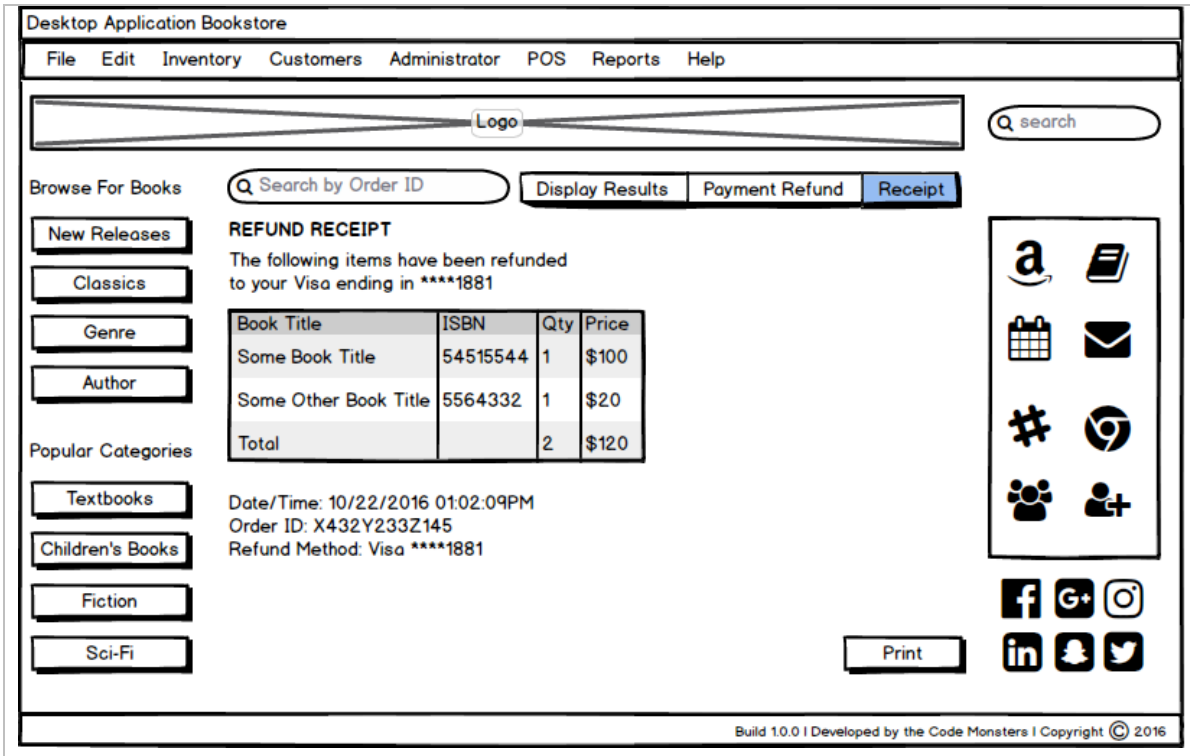

Figure 33: Processing Refund. Receipt.

4.7.4 ANALYSIS OBJECT MODEL

The Analysis Object Model displays the different Class Diagrams to be used within the application. The architecture used to create the program is the Model/View/Controller (MVC) architectural style. With this style, each of the subsystems are classified into one of three types of classes:

- Model
- View
- Controller

The Model contains all of the domain knowledge. This includes any type of processing of the code, retrieval of data from the database and insertion of data into the database. The View displays the content to the user. Any interaction between the user and the system will be handled by the Controller.

To simplify the subsystems, each View class is placed in the view directory. Similarly, each Controller class is placed into the controller directory and each Model class is placed into the model directory. The naming convention chosen is that each View class will end with “View” in its name. Each Controller will end with “Controller” in its name and each Model will end with “Model” in its name (for example, AddBookToInventoryController).

Rationale

The following diagrams were chosen to represent the desktop application software. The subsystems were modeled off of the Use Cases since they represented the system to its full potential. Figure 34 displays the main portion of the program. The different sections of the main screen have been broken down into further views called the CMPanelView and the MenuPanelView since they can be viewed as separate objects. Those objects meet to form the ApplicationWindowView.

Each subsystem was carefully designed to meet the needs of both the employee and the customer. Figures 37 and 38 represent the software’s reaction to a customer searching for a book by both keyword and by category; both are necessary functions that the customer must be able to do. Either way, the customer is capable of seeing the content on the screen. Once the user enters a keyword or clicks on a category the content is transferred via the controller to the model classes; the models then retrieve the applicable book data back to the controller and the controllers send the data back to the views. The views know how to manipulate the data and display the content as needed.

In order for the inventory to be displayed, orders have to be placed. A designated employee usually places the order and confirms its receipt. The following figures deal with the manipulation of inventory within the system:

- Figure 35: Adding a Book to the Inventory
- Figure 36: Deleting a Book from the Inventory
- Figure 39: Add Partial or Whole Order to Inventory
- Figure 40: Place new order
- Figure 41: Search for order both by Order ID and by ISBN of a book within an order
- Figure 46: Change the quantity of a particular book that’s currently in stock
- Figure 47: Modify the details of a particular book

Each of the above subsystems follows the same MVC logic. The employee is shown content through a view where he or she can interact with the subsystem. Upon an interaction with the subsystem, the controller is invoked to transfer the action to the model. The model retrieves or inserts the data and returns the results to the controller upon which the controller sends the data back to the view.

The system has two main types of users: employees and customers. The owner and/or administrator needs to be able to create additional employee accounts. Each of the employees need to be able to log into the system and terminate the session after each use so that another employee may access the terminal. It's also important to log out of a terminal to help prevent sensitive material, such as cost, from being displayed to a potential customer. The following figures deal with creation and manipulation of employee accounts as well as system access:

- Figure 42: Create any type of employee account, including administrative accounts
- Figure 44: Account Access. Logging in and out
- Figure 45: Resetting the user's password

The most crucial aspect of this software is for it to be able to accept payments. Each purchase goes through the following checkout process:

- Find items
- Get payment
- Print Receipt

The figures that deal with the checkout processes are:

- Figure 48: Point of Sales
- Figure 49: Process Refund

Finally, Figure 43 showcases the subsystem for generating sales and profit loss reports. The company needs to be able to view its current financial standing.

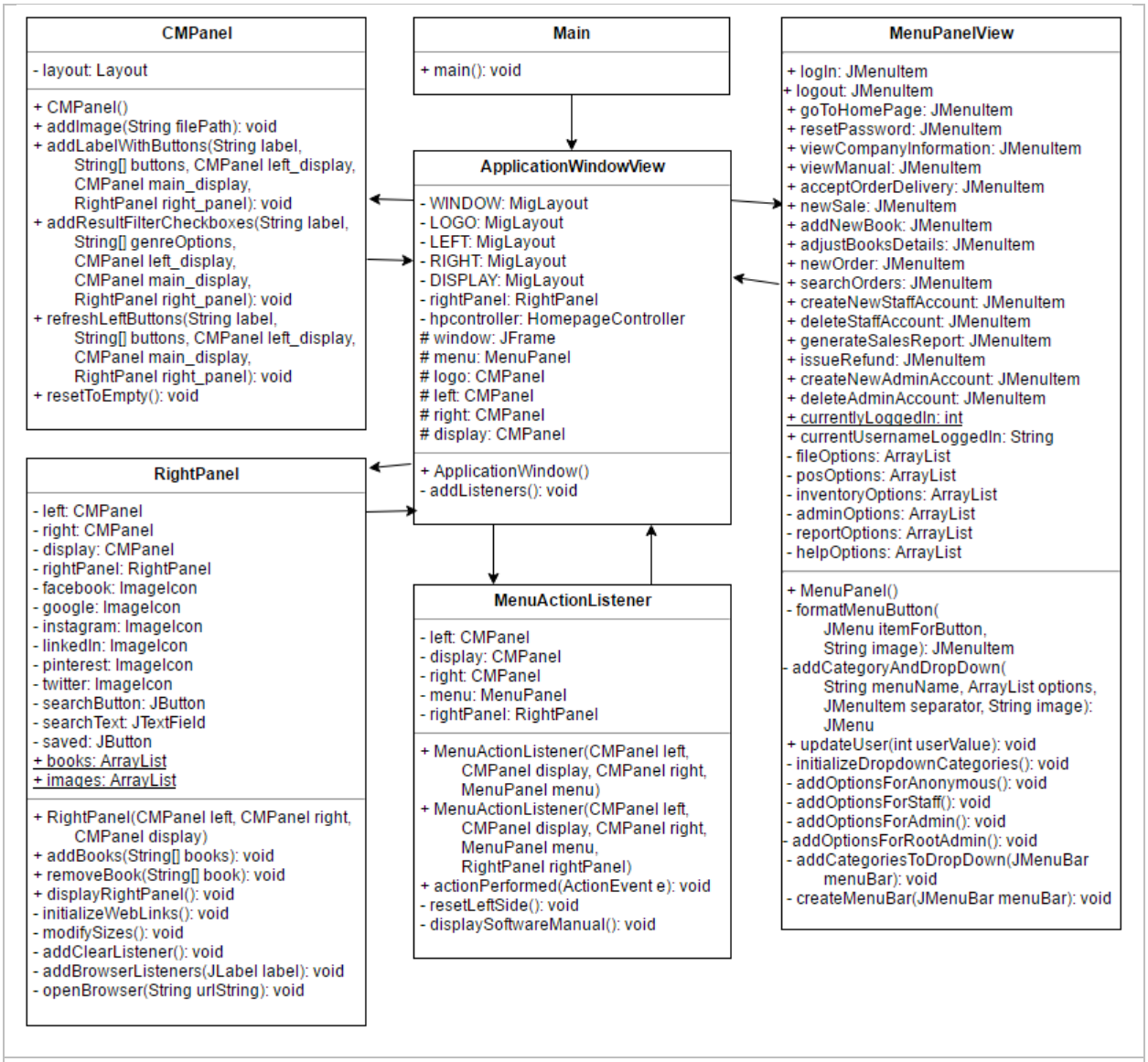

Figure 34: Initial Start.

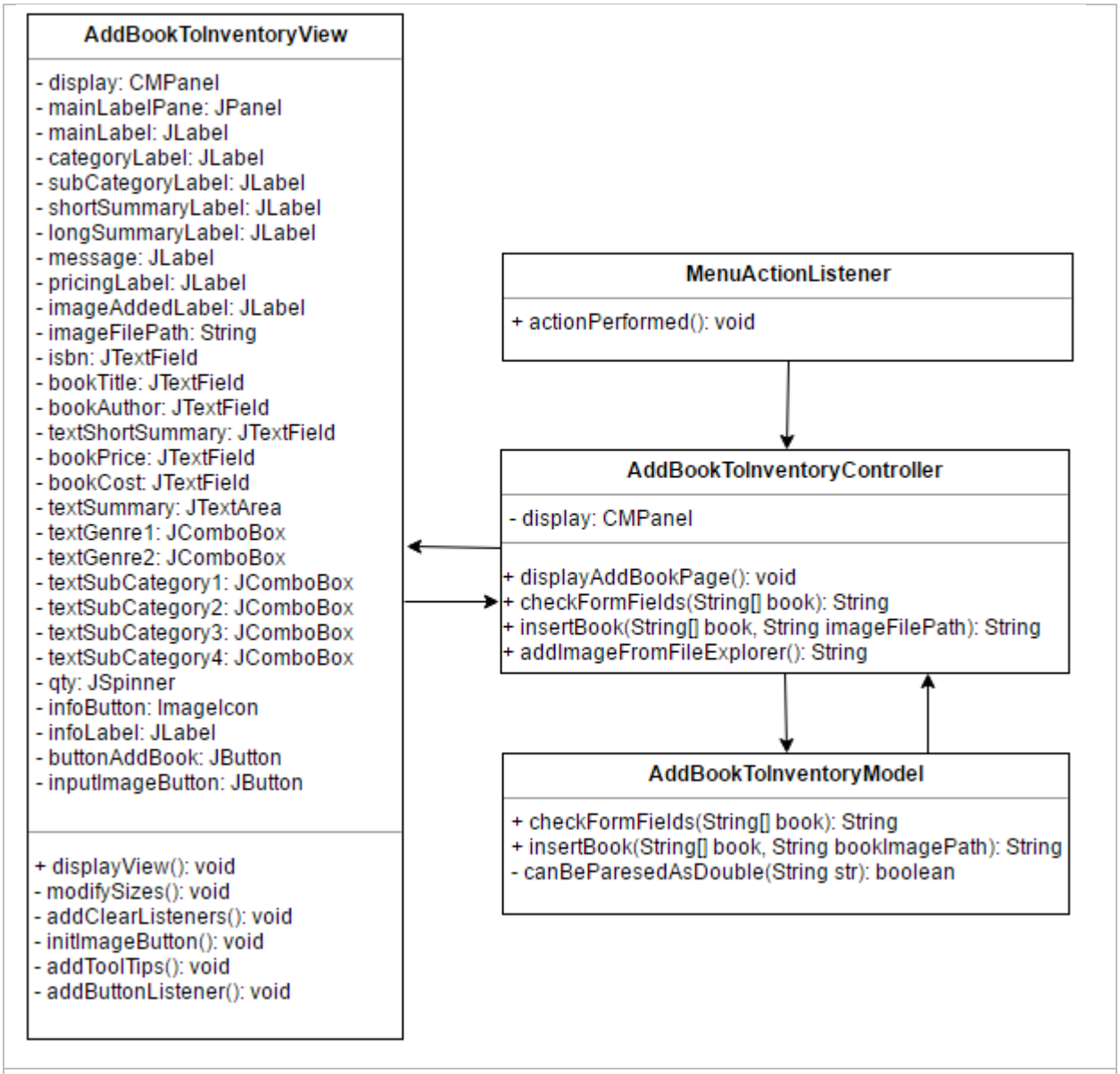

Figure 35: Adding a Book to the Inventory

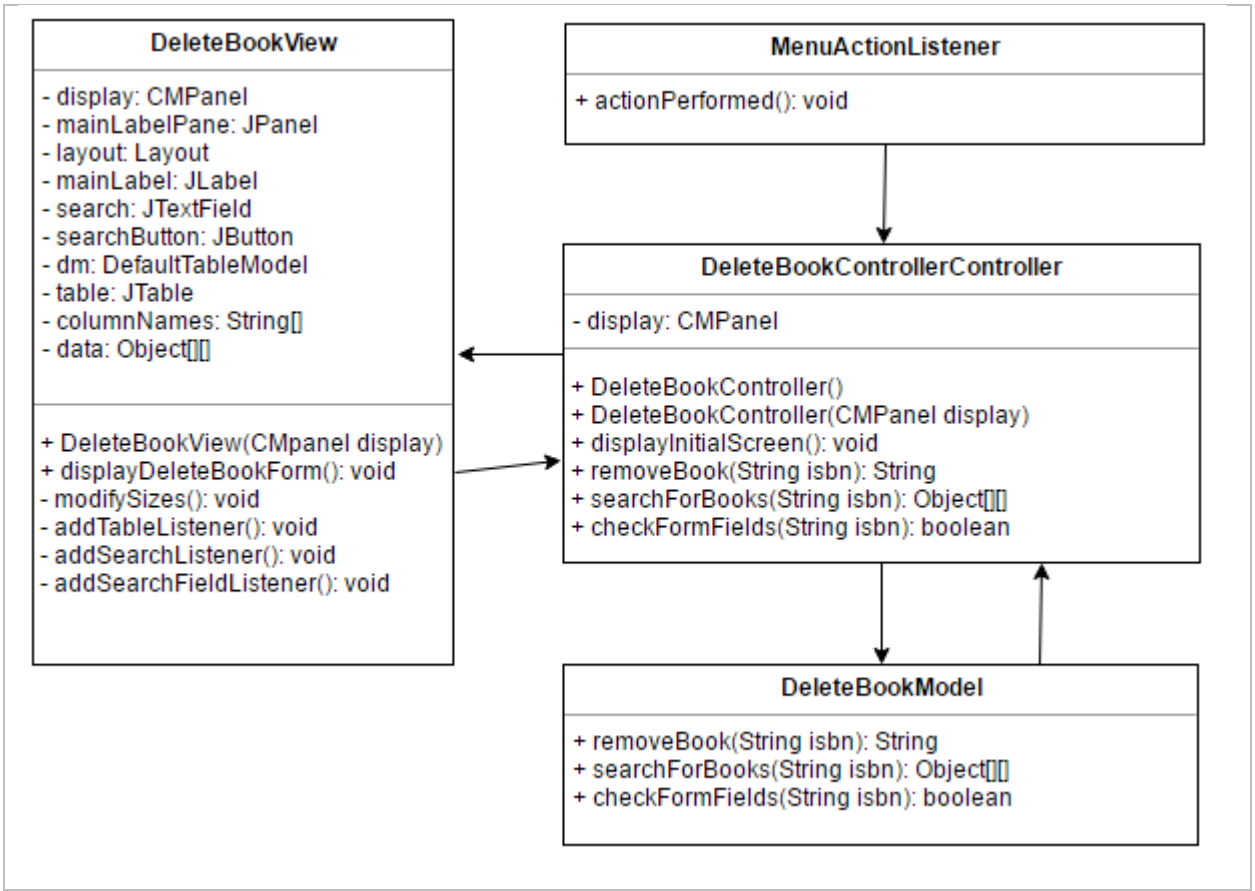

Figure 36: Deleting a Book from the Inventory

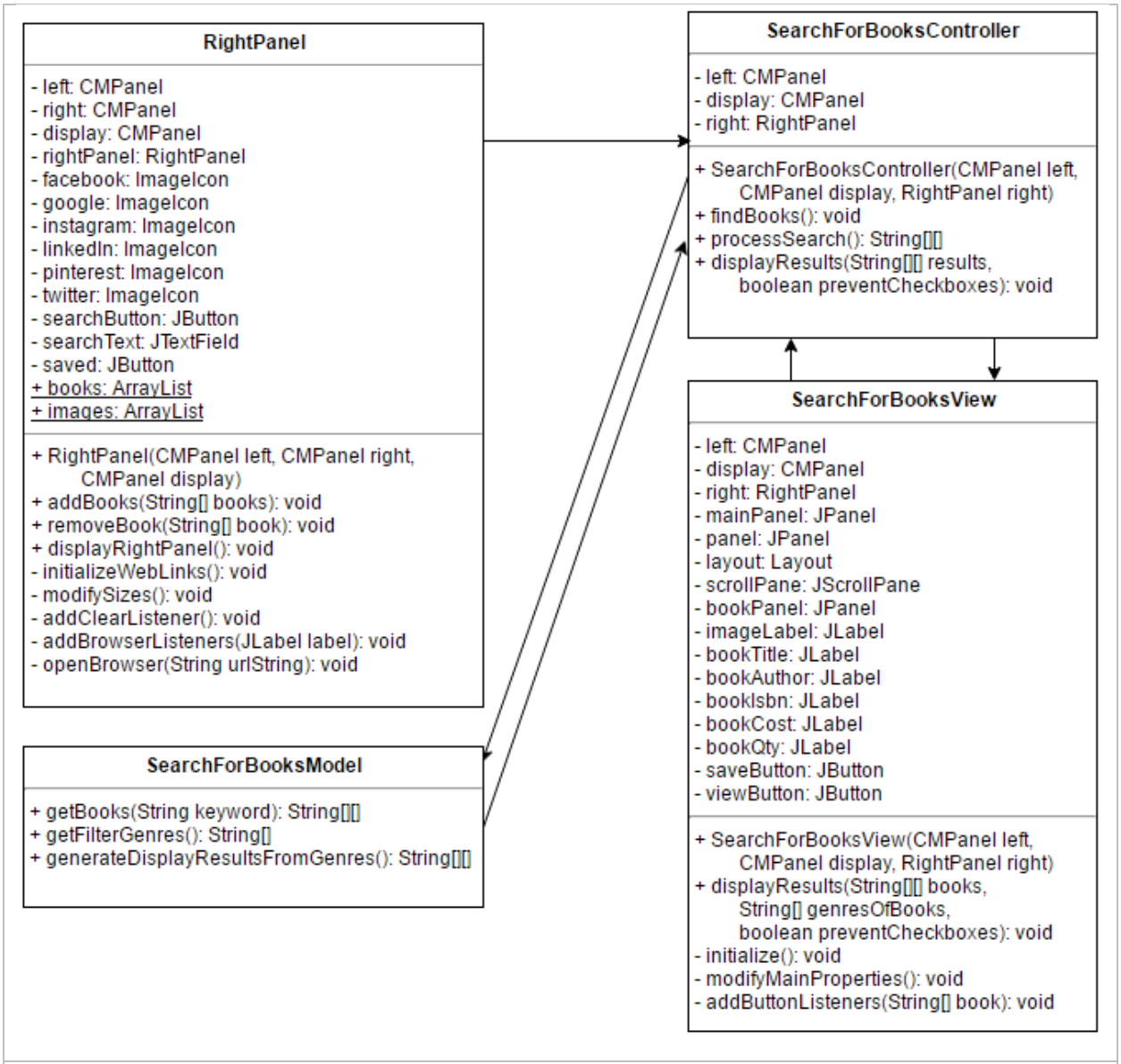

Figure 37: Searching for a Book

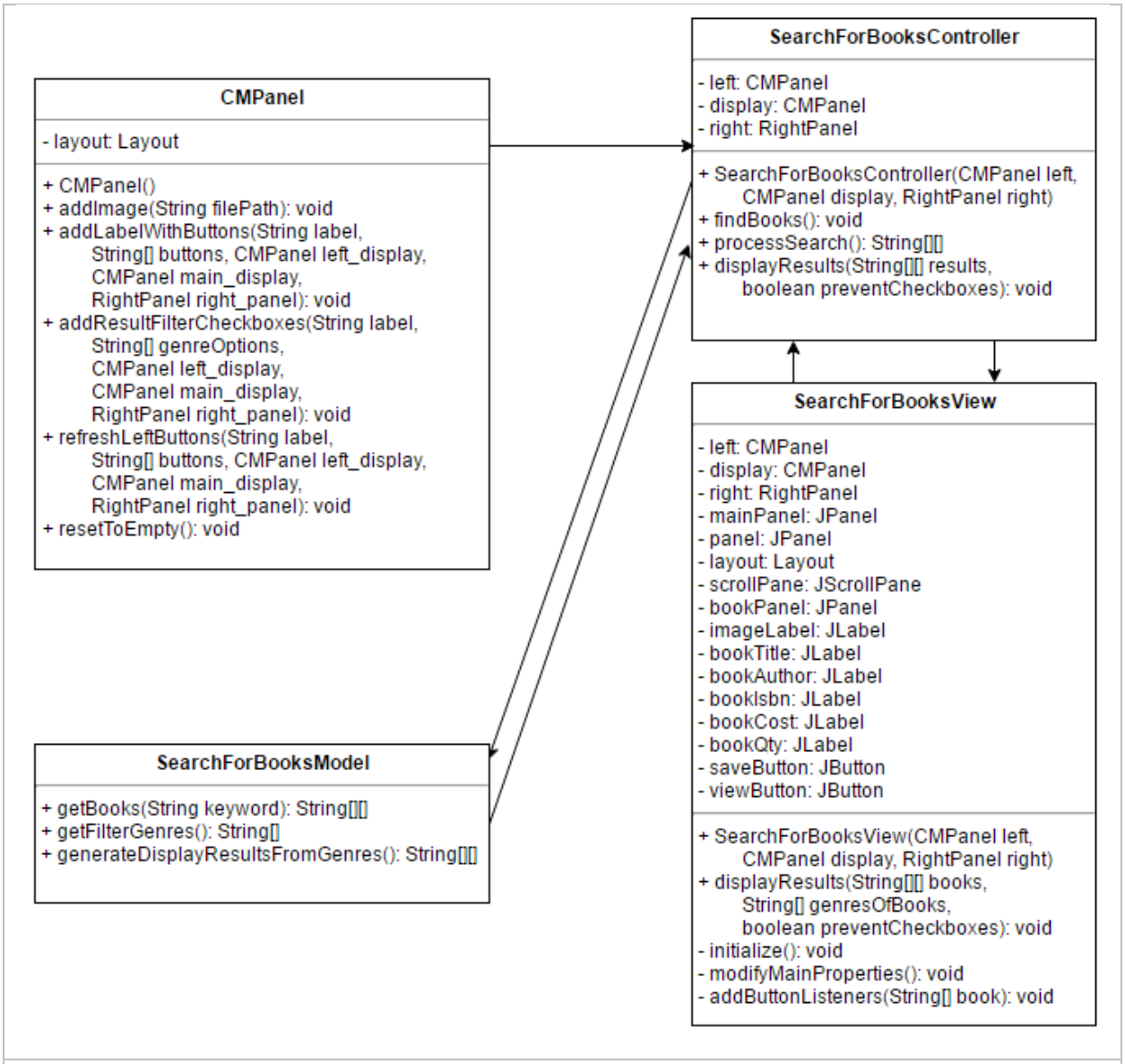

Figure 38: Browse for Book by Category

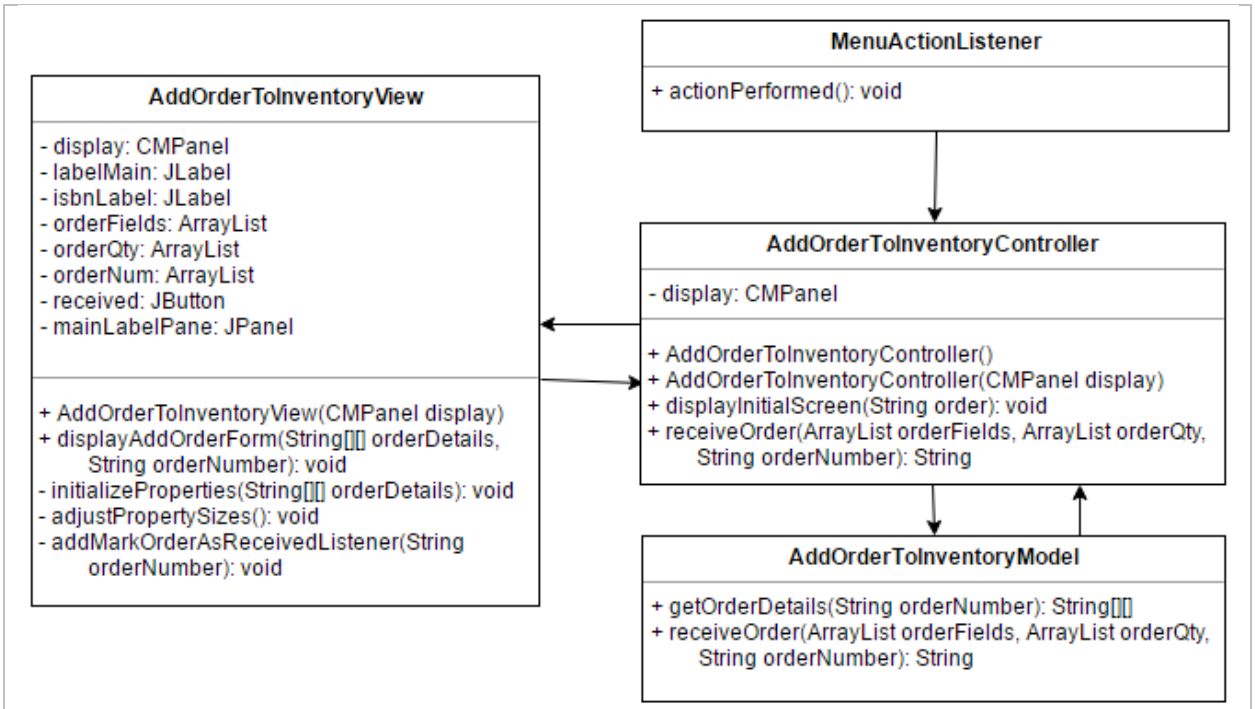

Figure 39: Add Partial or Whole Order to Inventory

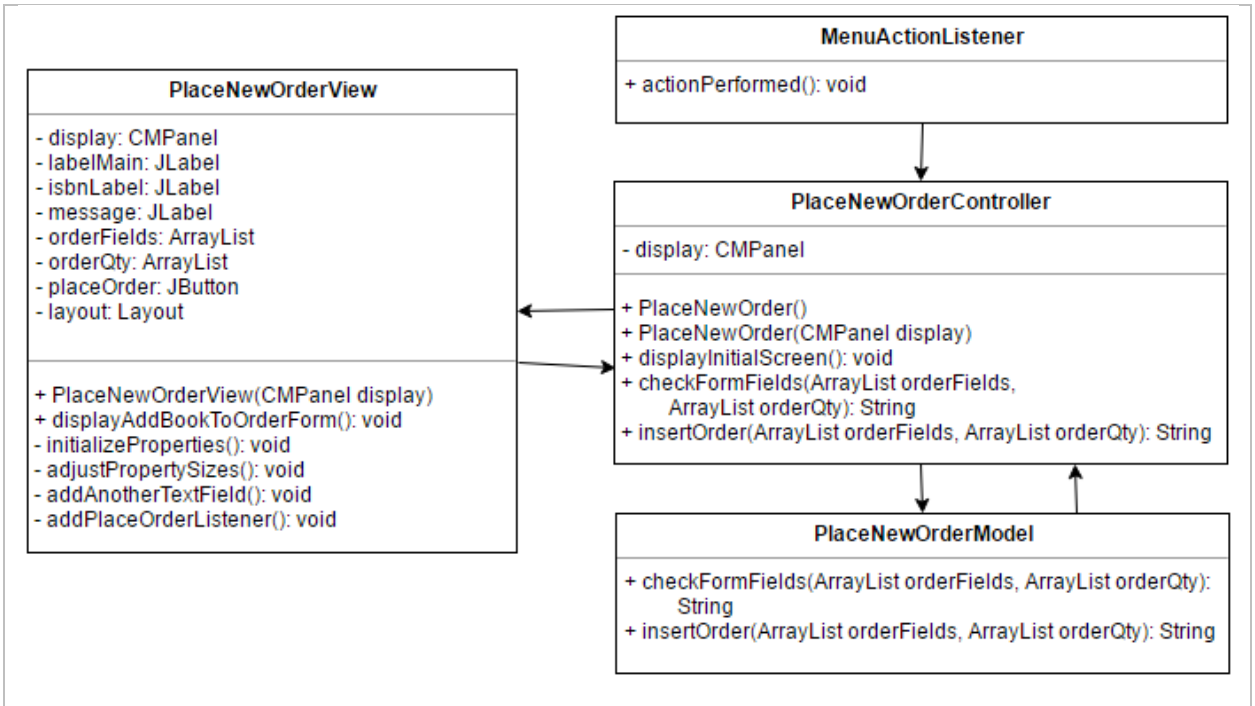

Figure 40: Place new order. Reference Use Case *PlaceNewOrder* for a detailed explanation.

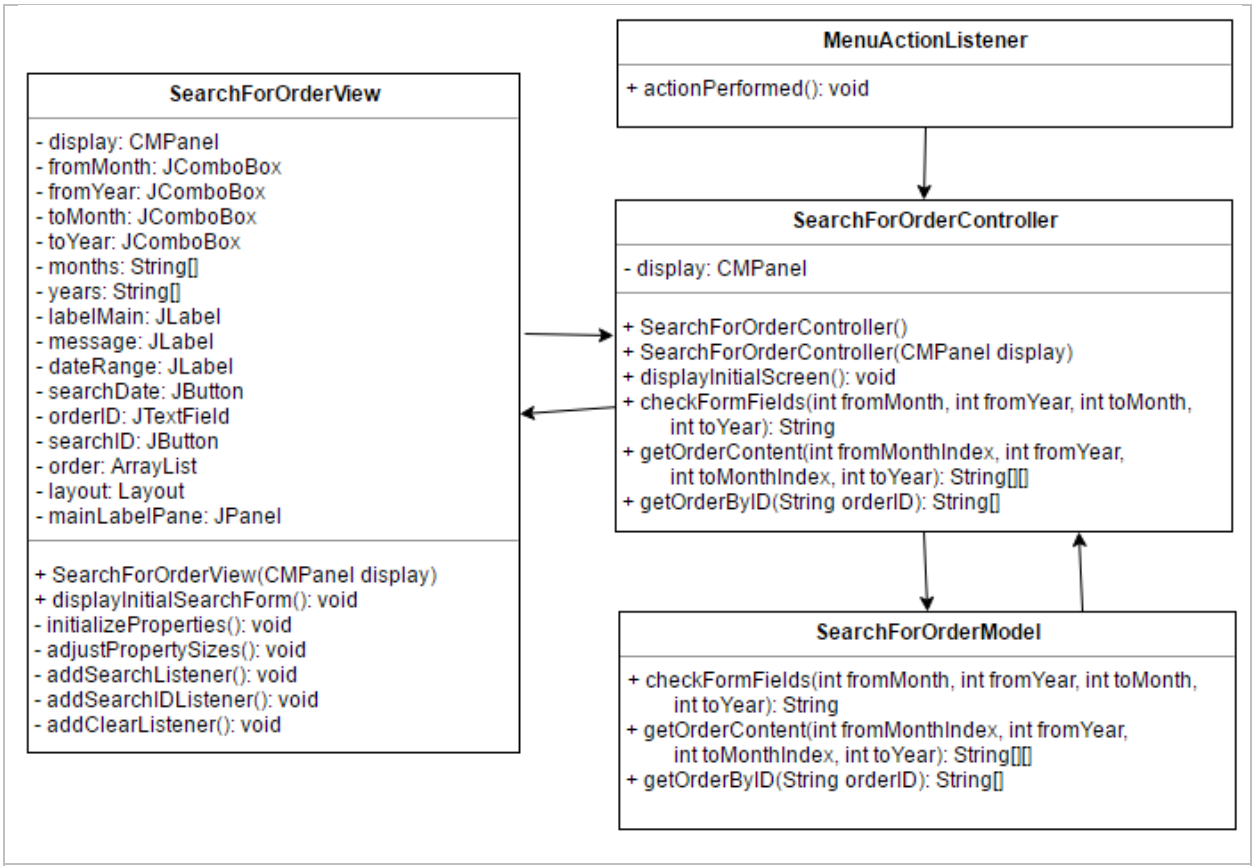

Figure 41: Search for order both by Order ID and by ISBN of a book within an order

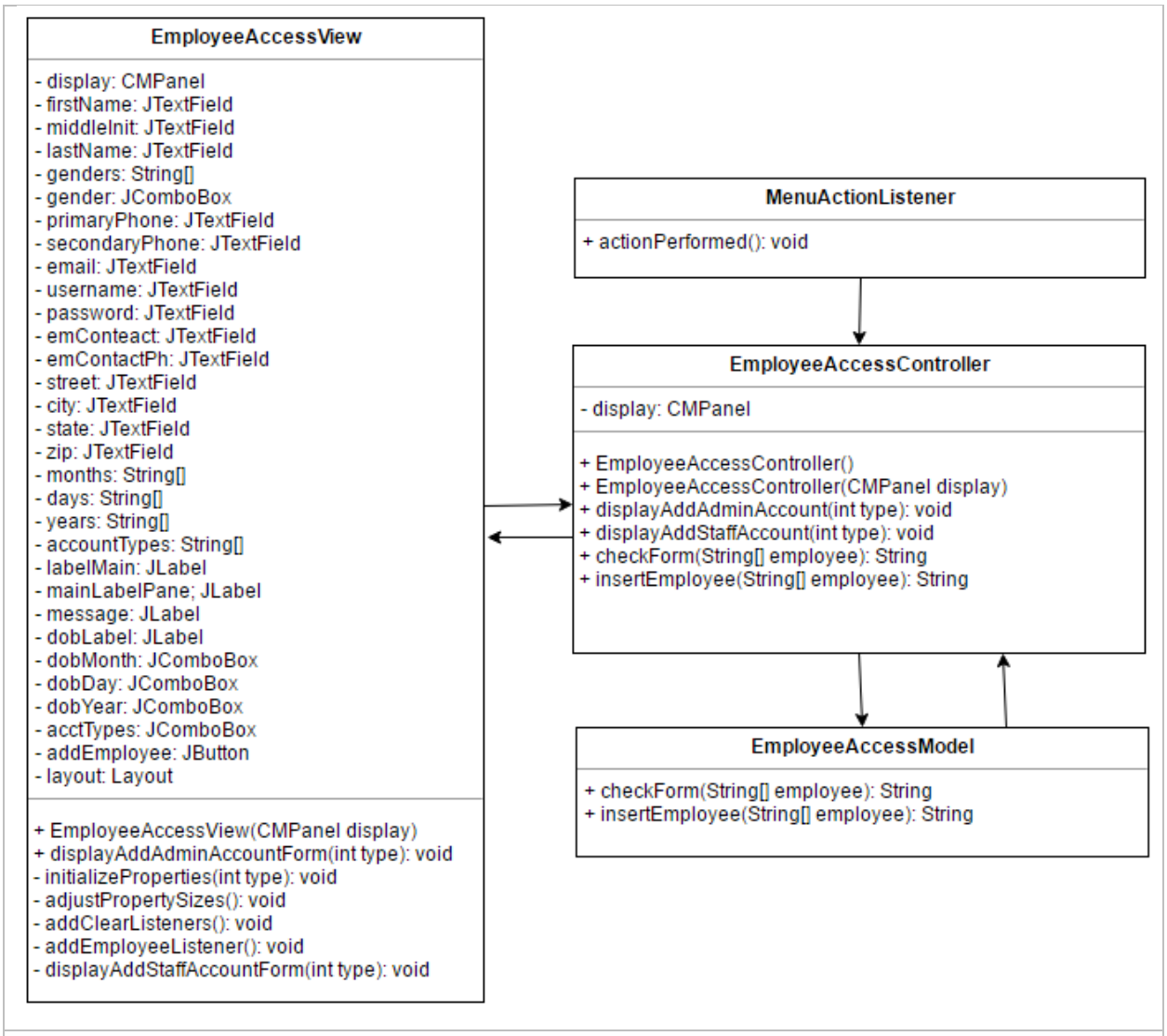

Figure 42: Create any type of employee account, including administrative accounts

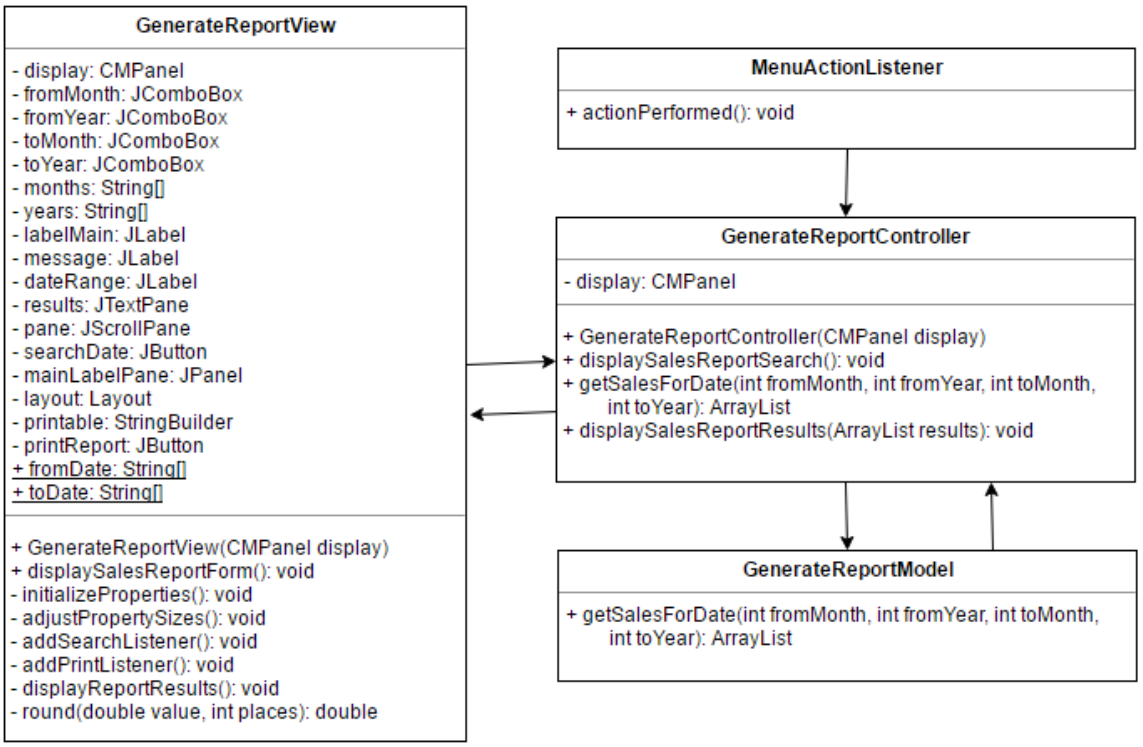

Figure 43: Generate Sales Report

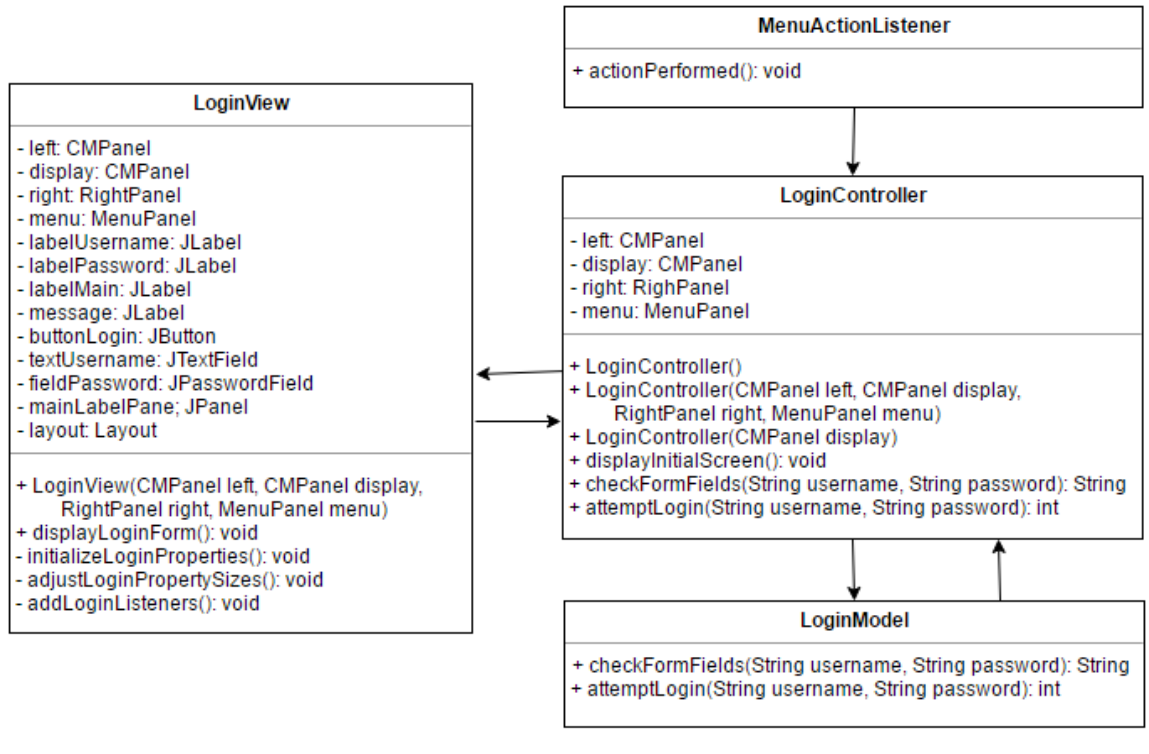

Figure 44: Account Access. Logging in and out.

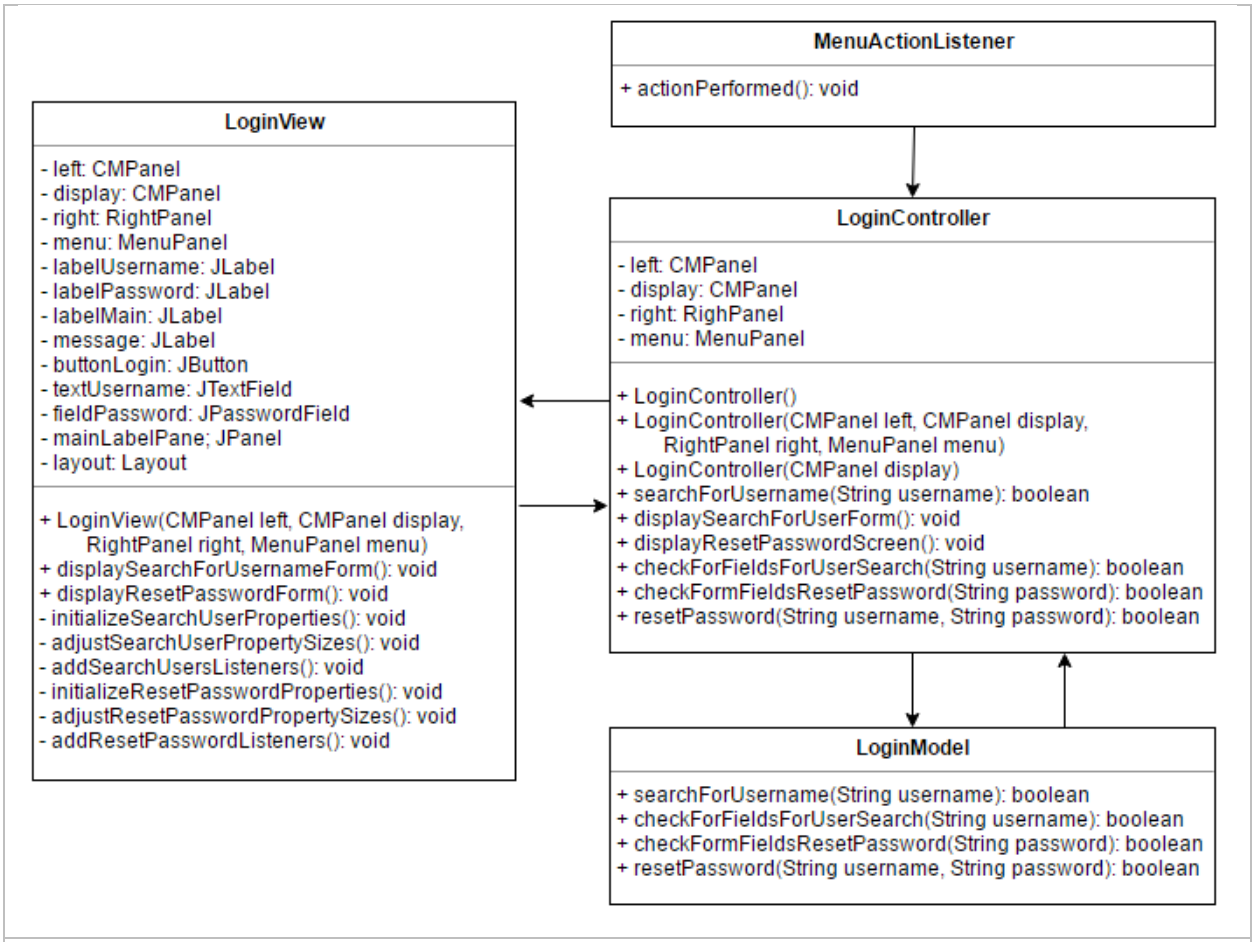

Figure 45: Resetting the user's password

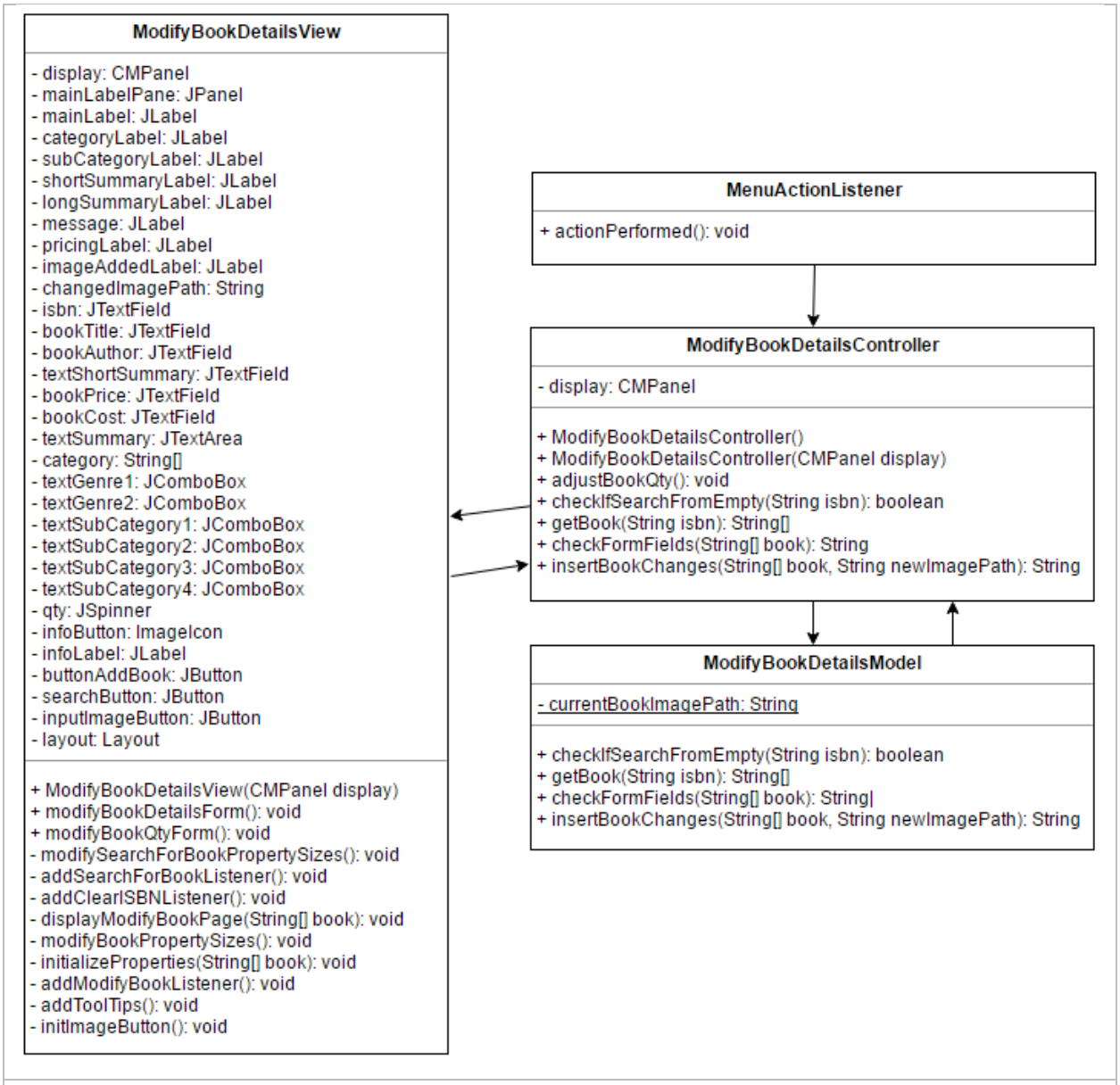

Figure 46: Change the quantity of a particular book that's currently in stock

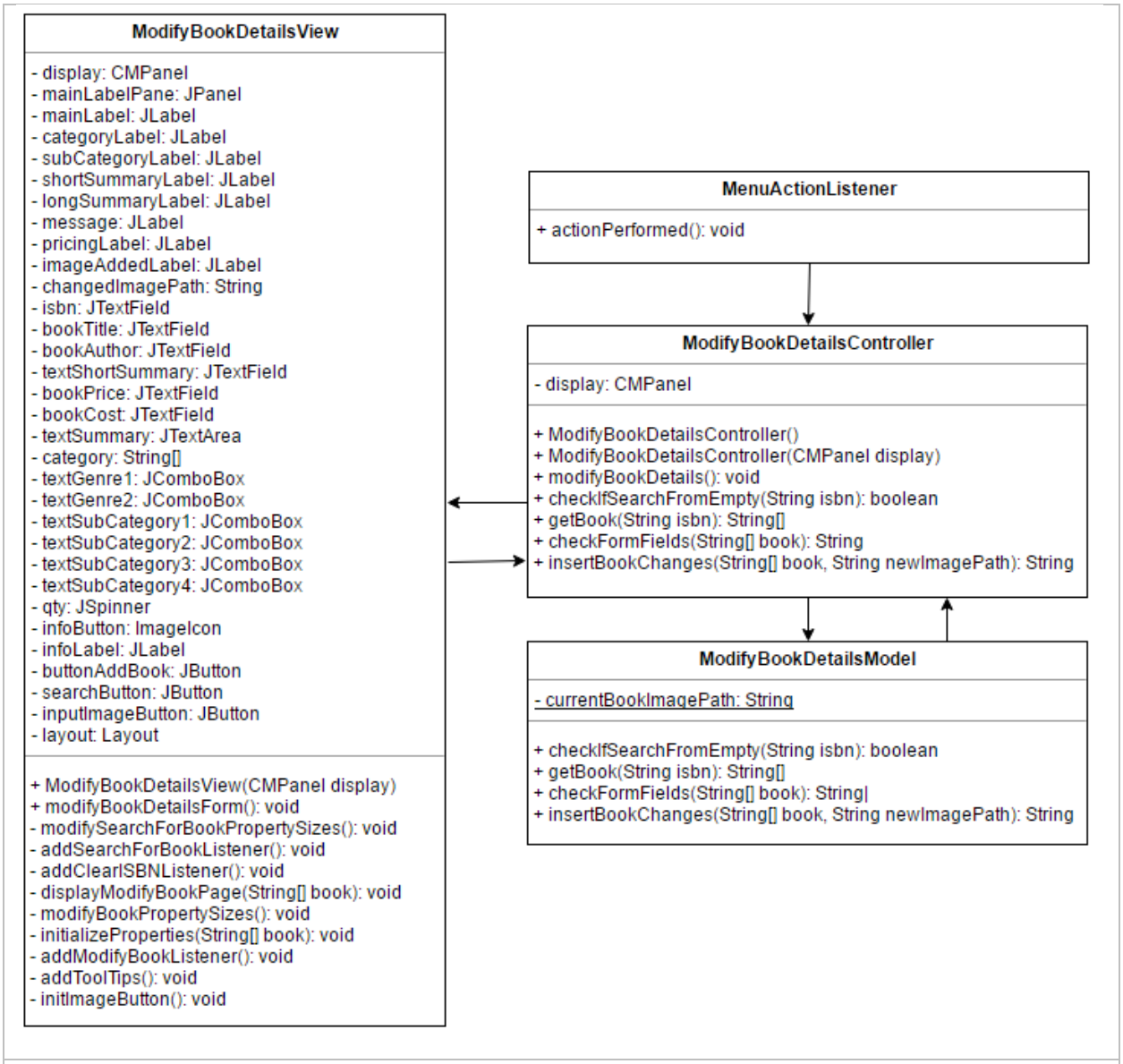

Figure 47: Modify the details of a particular book

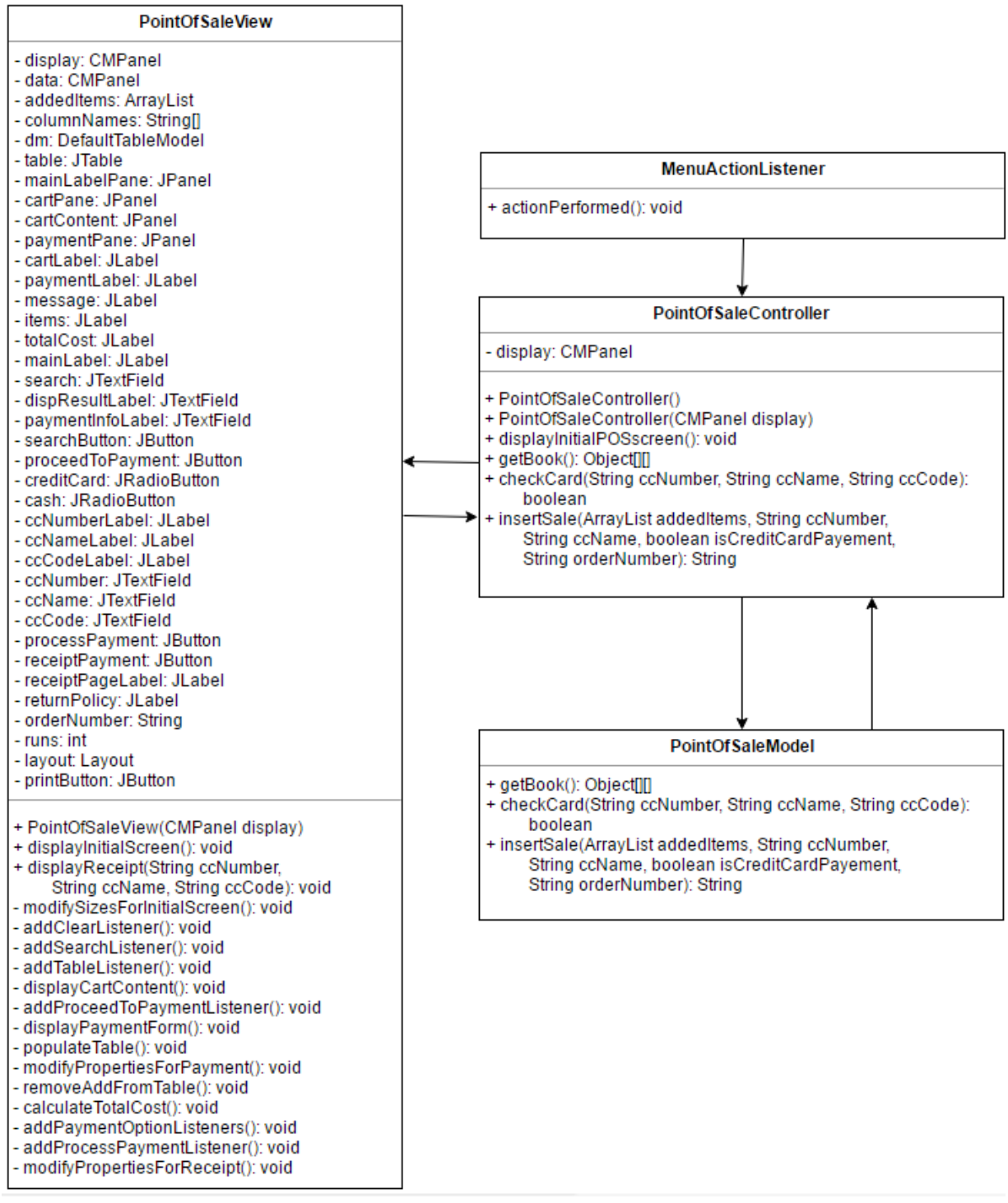

Figure 48: Point of Sales

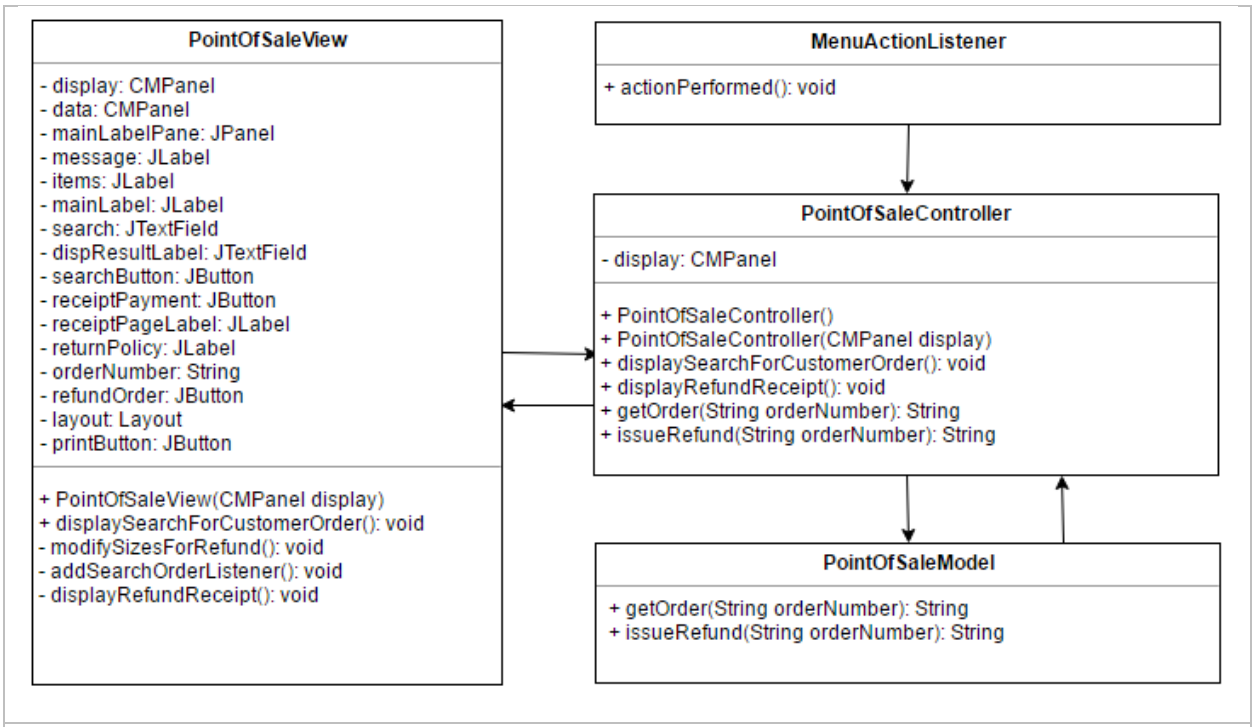

Figure 49: Process Refund

4.7.5 DYNAMIC MODEL (CATEGORY INTERACTION DIAGRAMS)

The dynamic model displays the visual representations of the interactions between categories that are required when implementing a Use Case.

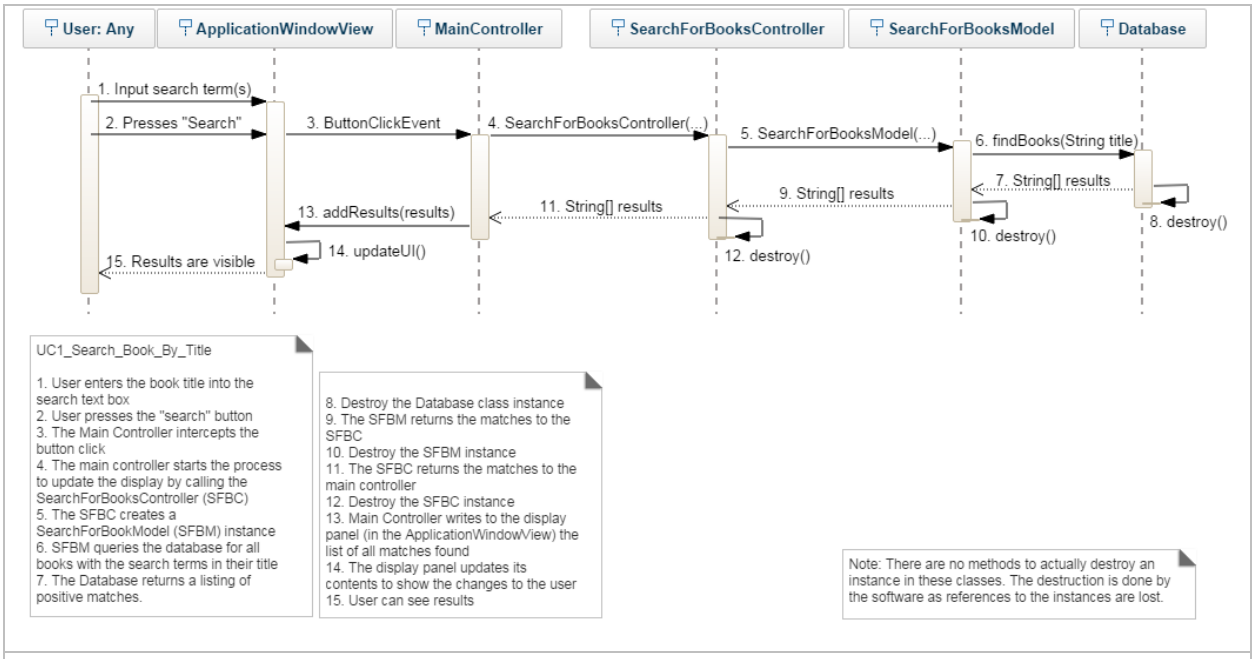

Figure 50: Search by Title Category Interaction Diagram

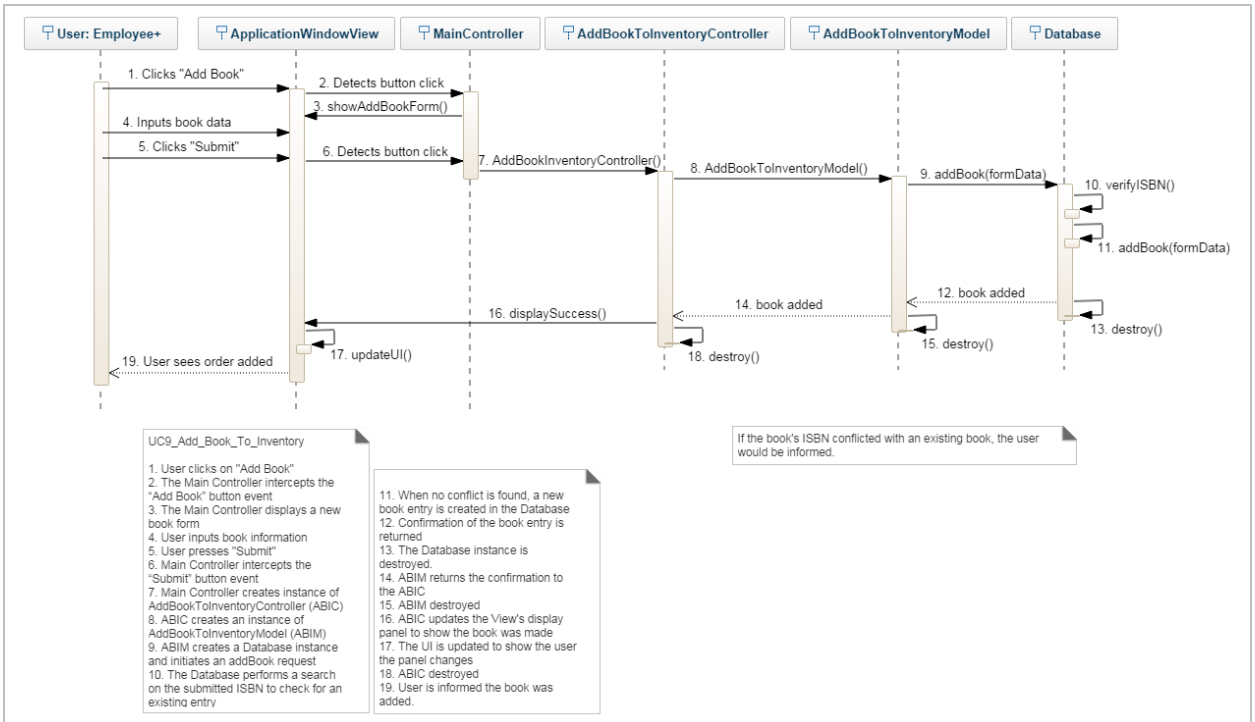

Figure 51: Add Book to Inventory Category Interaction Diagram

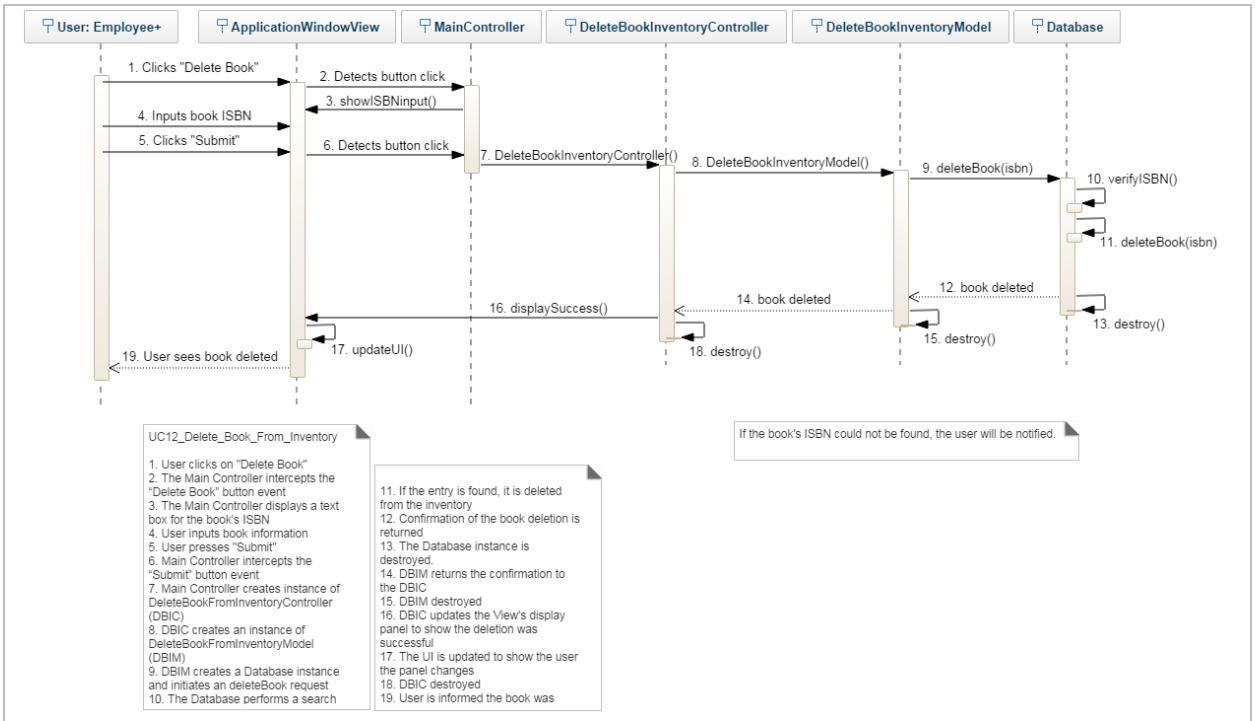

Figure 52: Delete Book from Inventory Category Interaction Diagram

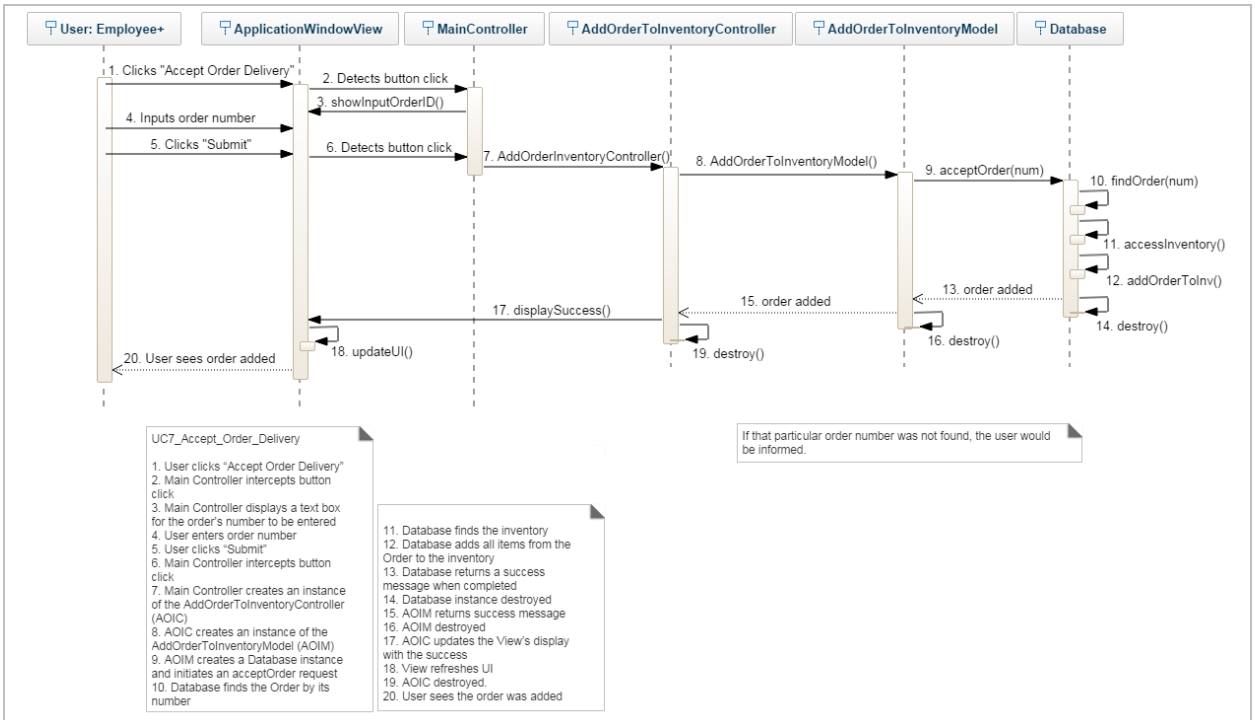

Figure 53: Accept Order to Inventory Category Interaction Diagram

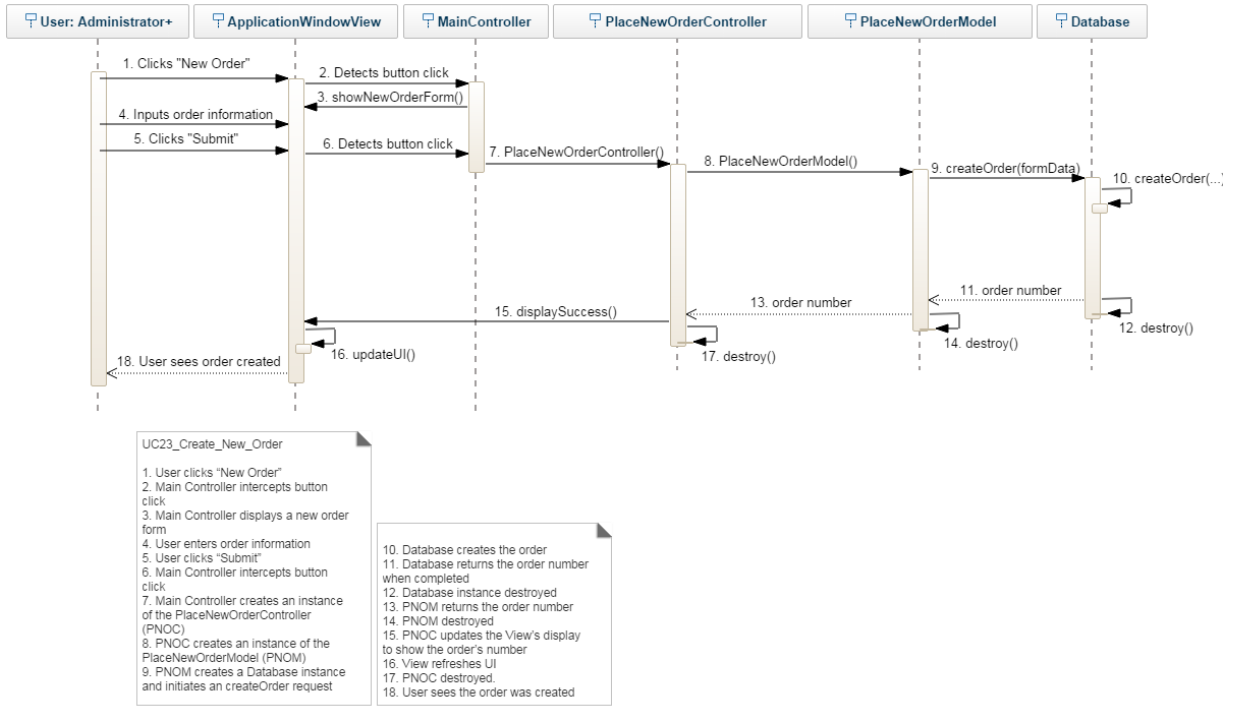

Figure 54: Create New Order Category Interaction Diagram

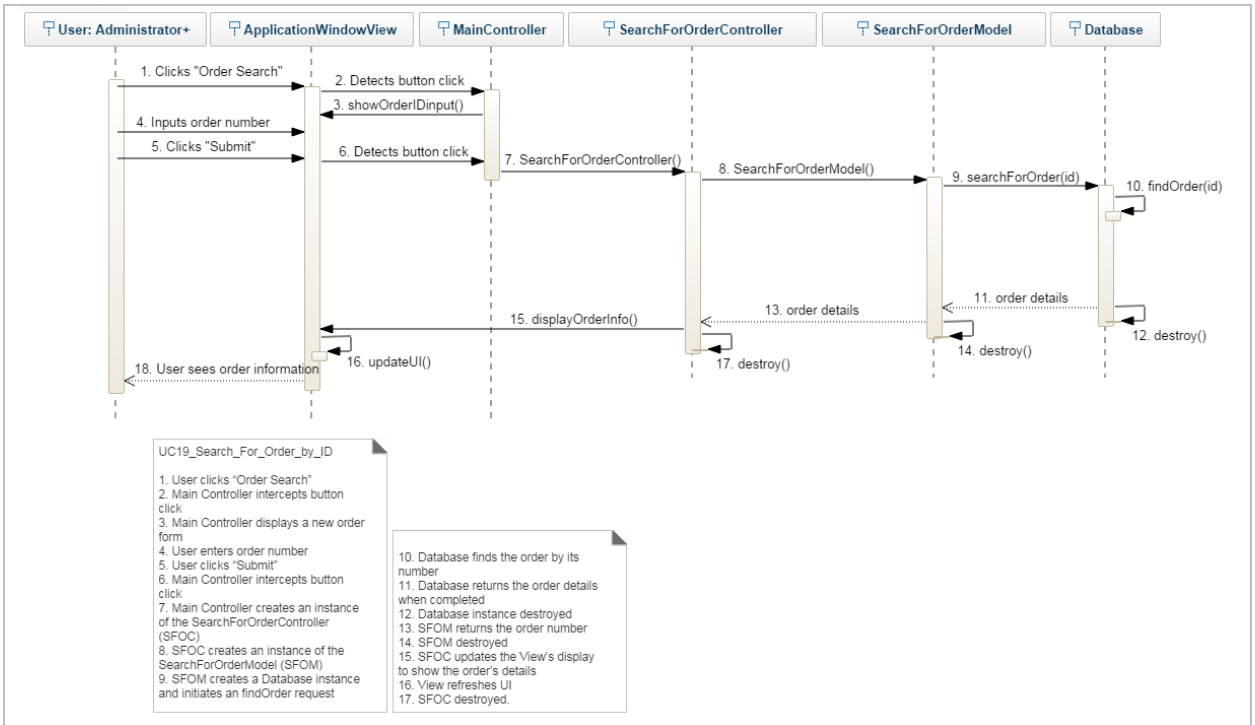

Figure 55: Search for Order Category Interaction Diagram

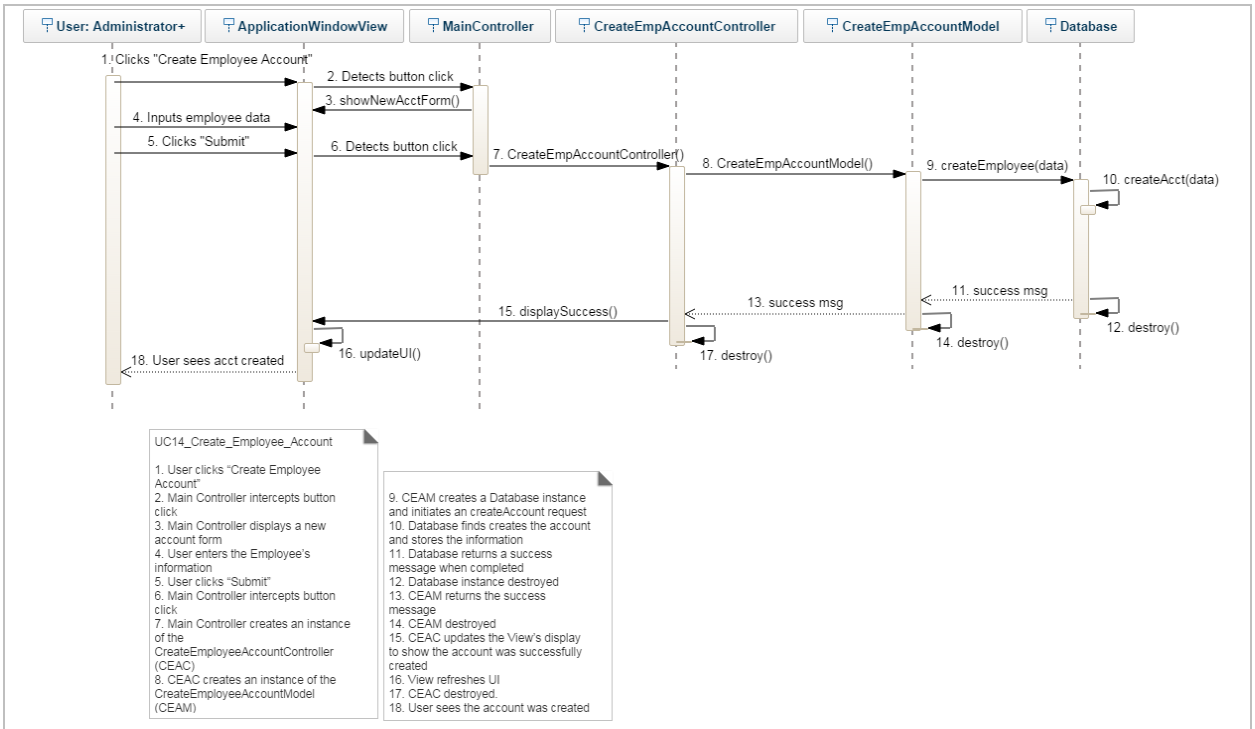

Figure 56: Create Employee Account Category Interaction Diagram

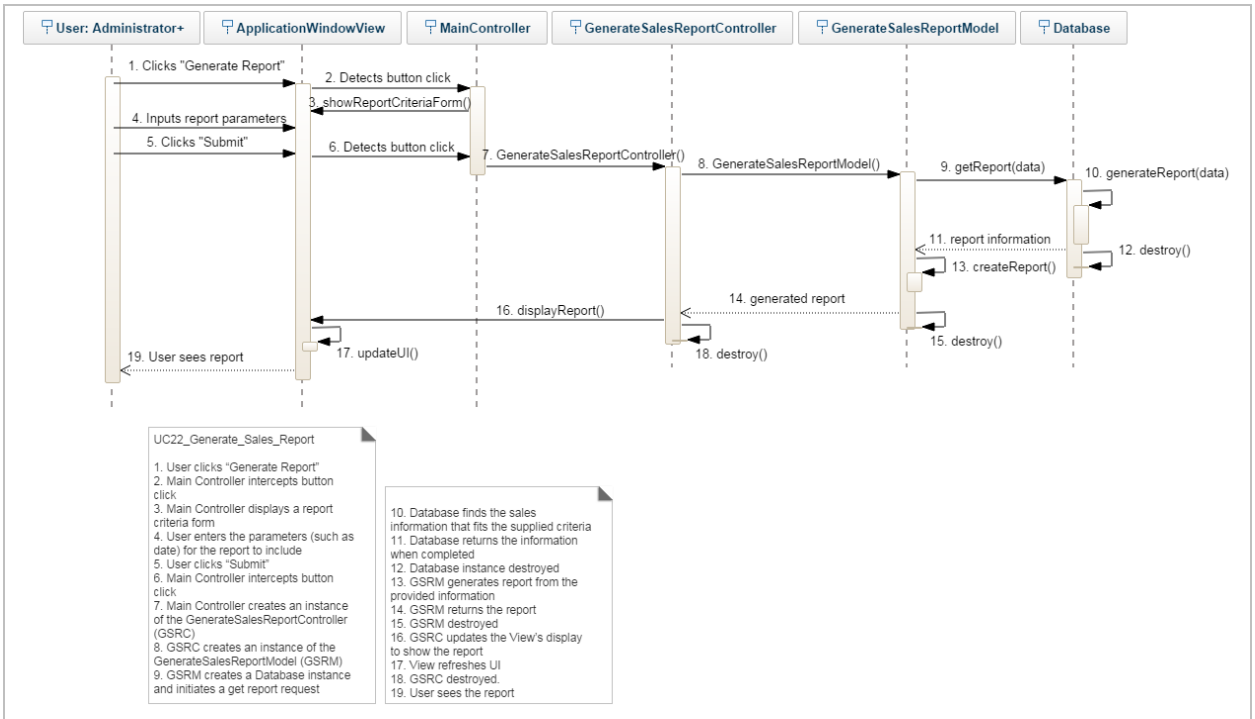

Figure 57: Generate Sales Report Category Interaction Diagram

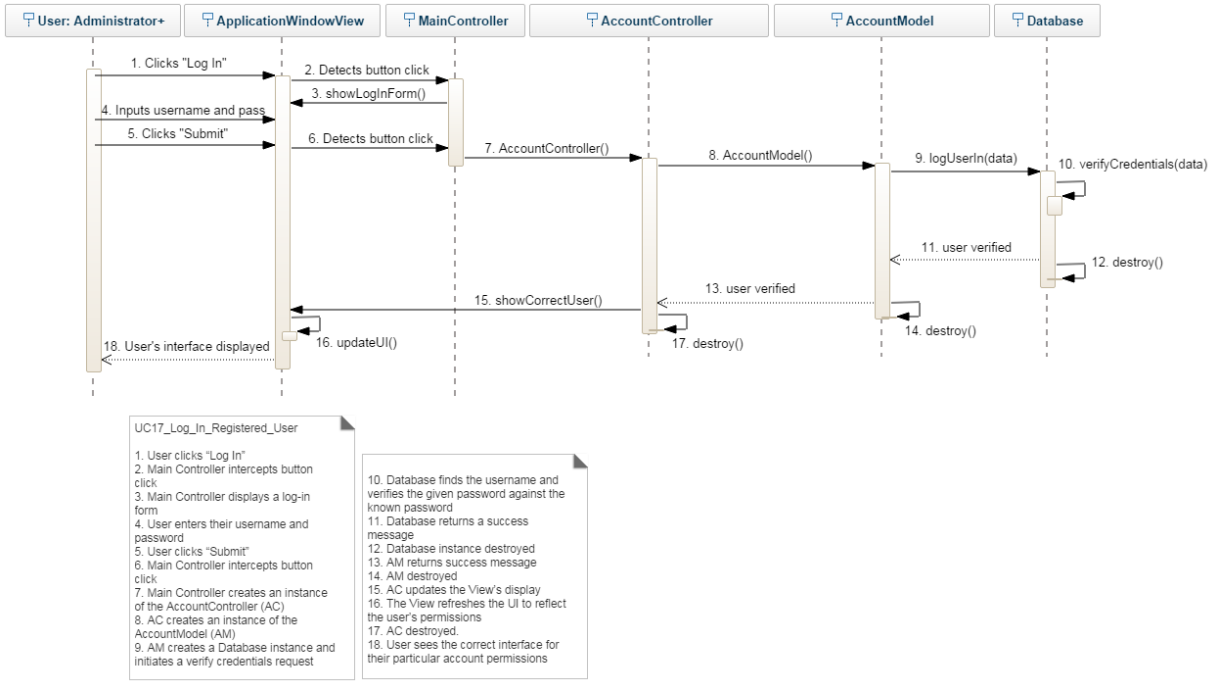

Figure 58: Logging into System Category Interaction Diagram

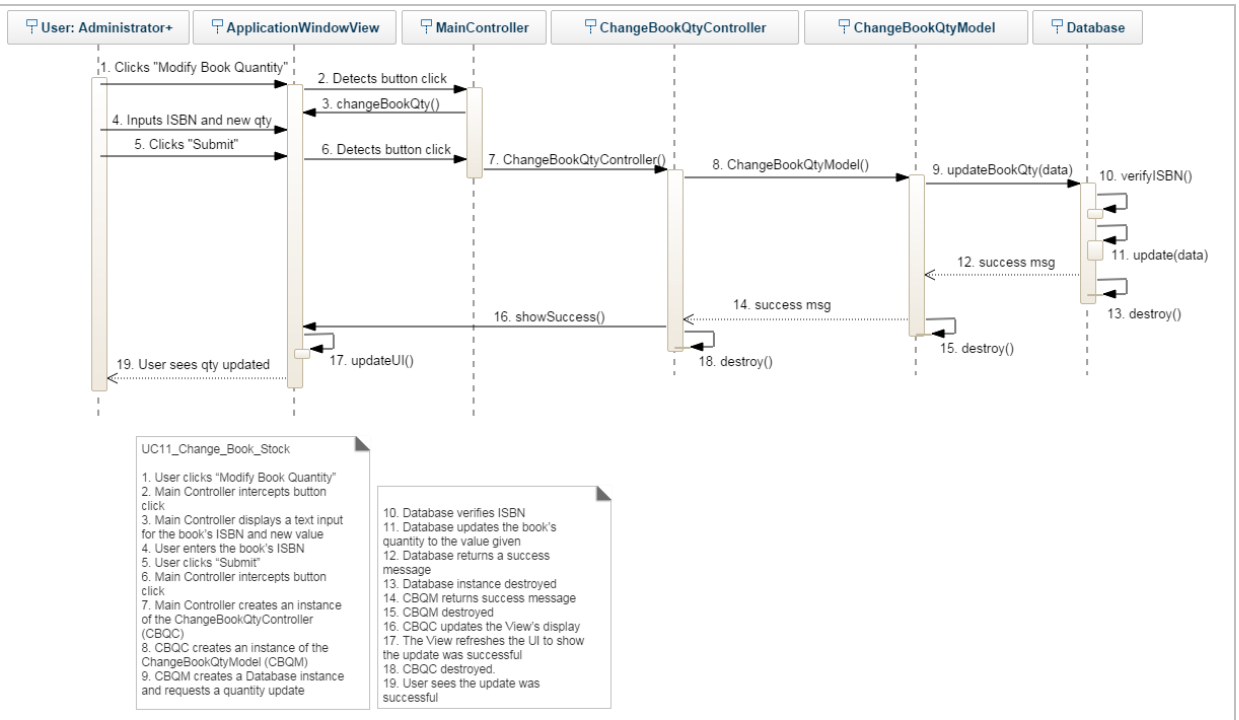

Figure 59: Change Book Stock Category Interaction Diagram

4.7.6 TEST CASES

Test-case identifier	**AddBookNotFromOrder_uc9**
Test location	./src/Model/AddBookToInventoryModel.java ./src/View/AddBookToInventoryView.java ./src/Controller/AddBookToInventoryController.java
Feature to be tested	Add a new book entity to the inventory
Feature Pass/Fail Criteria	The user should enter all the details for a new book to be added to the inventory. The application should accept only valid data. The inventory should reflect the addition nearly immediately after verified submission.
Means of control	Execute the application and test the feature as an Admin user: Log-in as Admin then begin the Test Procedure. When the procedure is completed, the Test is over.
Data / Results	11/4/2016 2:22pm **FAIL** IndexOutOfBoundsException encountered. Fixed in AddBookToInventoryModel class 11/4/2016 2:32pm **PASS** Book successfully added to inventory. Checks for duplicate ISBN, missing fields, and bad data-types all successful.
Test Procedure	With the application running, the user selects “Add Book To Inventory” and inputs all values on the form. When invalid values are submitted, the user should see a message describing the problem without losing their data entered thus far. When the user presses “Add Book” and all required information has been entered, a message will be displayed saying the book was added. If the inventory file is reviewed, the book should be in the inventory.
Rationale	Functional Requirement # 28
Special requirements	none

Test-case identifier	**ModifyBookDetails_uc10**
Test location	./src/Model/ModifyBookDetailsModel.java ./src/View/ModifyBookDetailsView.java ./src/Controller/ModifyBookDetailsController.java
Feature to be tested	Modify the details of a single entity in the inventory
Feature Pass/Fail Criteria	The user should see the book details as represented in the inventory file after searching for the book's ISBN. Changes made should be immediately represented in the inventory file.
Means of control	Execute the application and test the feature as an Admin user: Log-in as Admin then begin the Test Procedure. When the procedure is completed, the Test is over.
Data / Results	11/4/2016 2:41pm **FAIL** Form not properly populating with Sub-Categories. Sub-Categories that are not on the hard-coded list ("Fiction", "Non-Fiction" presently) will not display. This causes an infinite-loop where the existing sub-categories cannot be preserved. Changes to the ModifyBookDetailsView/Controller will be required. Additionally, the current View does not include fields for "price" and "cost" which cause exceptions due to null data fields. **UNRESOLVED**. 11/13/2016 11:45am **PASS** Static constants in the Config file are used to populate all dropdown menus that display book genres. Now all existing books in the inventory will show on the form as intended.
Test Procedure	With the application running, the user selects "Adjust A Book's Details". The user will search for a book by its ISBN. If the ISBN is not found, the user is notified by a message. If it is found, the book's details are pre-filled into a form and displayed to the user. The user inputs the desired information and presses "Modify Details". If all changes are valid, the book will be updated in the inventory file, otherwise appropriate messages will be displayed to the user. If changes are accepted, the inventory file will be immediately updated.
Rationale	Functional Requirement # 29
Special requirements	The inventory items should satisfy all constraints necessary to be added as a new book using the application. Books that are manually added to the inventory for testing may circumvent these constraints and therefore cause errors/exceptions (unhandled).

Test-case identifier	**DeleteBookFromInventory_uc12**
Test location	./src/Model/DeleteBookModel.java ./src/View/DeleteBookView.java ./src/Controller/DeleteBookController.java
Feature to be tested	Delete an entire entry from the inventory
Feature Pass/Fail Criteria	The user should see the correct book details after searching by the ISBN. When deleted, the book entry should be entirely removed from the inventory without effecting other entries of the inventory.
Means of control	Execute the application and test the feature as an Admin user: Log-in as Admin then begin the Test Procedure. When the procedure is completed, the Test is over.
Data / Results	11/3/2016 9:54pm **FAIL** No messages were being displayed on invalid ISBNs. Attempting to delete an invalid result erased the entire inventory. 11/3/2016 10:10pm **FAIL** No messages were being displayed on invalid ISBNS. 11/3/2016 10:16pm **PASS** Messages are properly displayed and the correct, unique instance of the book is removed from the inventory after multiple tests.
Test Procedure	With the application running, the user selects “Delete a Book”. The user must enter a book’s ISBN. Searching will display the book’s details in the table if the ISBN is valid. If the ISBN is not valid, a message will be displayed instead. Clicking on the “Delete?” column of the book entry will remove the book from the inventory. Checking the inventory file should show the book entity is permanently removed.
Rationale	Functional Requirement # 30
Special requirements	The ISBN must match exactly. No partial-match-searches are possible at this time.

Test-case identifier	**ResetUserPasswordAnyUser_uc18**
Test location	./src/Model/LoginModel.java ./src/View/LoginView.java ./src/Controller/LoginController.java
Feature to be tested	User changes their own password
Feature Pass/Fail Criteria	A user should be able to change the password for a given username if they know the current password for that username. The change should be represented in the credentials' file immediately.
Means of control	Execute the application then begin the Test Procedure. When the procedure is completed, the Test is over.
Data / Results	11/3/2016 10:40pm **FAIL** messages saying the username was not found are displayed even for valid usernames. Changes to View corrected this bug. 11/3/2016 10:46pm **FAIL** No prompt for the current username's password is being displayed. **UNRESOLVED** 11/10/2016 10:00am **PASS** Form properly shows both fields. Matches return a "success" message in the logs which is enough to move forward and add on to this functionality.
Test Procedure	With the application running, the user clicks "Log-in". Click on "Reset Password". Input the desired username and press "Search". If the username is not found, an appropriate message will be displayed. If the username is found, input the existing password then the new password. If the existing password is not correct, no change will be made. If it is correct, the change should be stored in the username credentials' file immediately.
Rationale	Functional Requirement # 37
Special requirements	none

Test-case identifier	**ResetUserPasswordAdminOnly_uc18**
Test location	./src/Model/LoginModel.java ./src/View/LoginView.java ./src/Controller/LoginController.java ./src/GUIObjects/MenuPanel.java
Feature to be tested	Admin user resets any user's password to a custom password of their choice.
Feature Pass/Fail Criteria	An admin-level user changes any other user's (except "root") password to a custom, non-whitespace-only string. The change should be represented in the credentials' file immediately.
Means of control	Execute the application, Log-in as an Administrator then begin the Test Procedure. When the procedure is completed, the Test is over.
Data / Results	11/3/2016 11:11pm **FAIL** The root's password can be changed. Test needs to be added in the Model. **UNRESOLVED** 11/23/2016 12:44pm **PASS** A change to the LoginModel.java and MenuPanel.java conducts a check that the user trying to make the change is of equal or higher level than the user being changed.
Test Procedure	With the application running, the user clicks "Log-in". Click on "Reset Password". Input the desired username and press "Search". If the username is not found, an appropriate message will be displayed. If the username is found, input the existing password then the new password. If the existing password is not correct, no change will be made. If it is correct, the change should be stored in the username credentials' file immediately.
Rationale	Functional Requirement # 37
Special requirements	none

Test-case identifier	**CreateAdminAccount_uc16**
Test location	./src/Model/EmployeeAccessModel.java ./src/View/EmployeeAccessView.java ./src/Controller/EmployeeAccessController.java ./src/Listeners/MenuActionListener.java ./src/GUIobjects/MenuPanel.java
Feature to be tested	A Root user can create a new administrator account.
Feature Pass/Fail Criteria	Without over-writing an existing account or deleting an existing account, create a new Adminstrator-level account with a unique username and email.
Means of control	Execute the application, Log-in as Root then begin the Test Procedure. When the procedure is completed, the Test is over.
Data / Results	11/4/2016 8:10pm **FAIL** Account information is not being assigned to the correct keys in the database. "Username" is pointing to "Address" and various other mistakes. Fixed: Wrong array was passed in a parameter causing improperly formatted data to be parsed for the account. 11/4/2016 8:48pm **FAIL** Checks for unique usernames and email addresses are encountering a parse error and EOF errors. Fixed: The file should have at least one valid account in it to be compatible with the current logic of the database. Manually inserted a Root account to overcome EOF error 11/4/2016 9:12pm **PASS** All form validations and existing account checks are successful.
Test Procedure	With the application running, click on "Create New Administrator Account". A form to create a user will be shown. Fill in all the necessary fields and press "Create". If any required fields are left blank or if the username / email address are already used by other users, a message will notify you of the specific problem. When no problems are found, a "Success" message should be displayed. At this time, the new account should exist in the credentials file.
Rationale	Functional Requirement # 34
Special requirements	Can simulate being logged in as Root without actually having a root account in the file. If the credentials file is empty, parsing exceptions may occur causing the process to fail.

Test-case identifier	**CreateStaffAccount_uc14**
Test location	./src/Model/EmployeeAccessModel.java ./src/View/EmployeeAccessView.java ./src/Controller/EmployeeAccessController.java ./src/Listeners/MenuActionListener.java ./src/GUIobjects/MenuPanel.java
Feature to be tested	An Admin or Root user can create a new employee/staff account.
Feature Pass/Fail Criteria	Without over-writing an existing account or deleting an existing account, create a new employee-level account with a unique username and email.
Means of control	Execute the application, Log-in as Root then begin the Test Procedure. When the procedure is completed, the Test is over.
Data / Results	11/4/2016 9:18pm **PASS** All form validations and existing account checks are successful.
Test Procedure	With the application running, click on “Create New Staff Account”. A form to create a user will be shown. Fill in all the necessary fields and press “Create”. If any required fields are left blank or if the username / email address are already used by other users, a message will notify you of the specific problem. When no problems are found, a “Success” message should be displayed. At this time, the new account should exist in the credentials file.
Rationale	Functional Requirement # 33
Special requirements	If the credentials file is empty, parsing exceptions may occur causing the process to fail.

Test-case identifier	**DeleteStaffAccount_uc15**
Test location	./src/Model/EmployeeAccessModel.java ./src/View/EmployeeAccessView.java ./src/Controller/EmployeeAccessController.java ./src/Listeners/MenuActionListener.java ./src/GUIobjects/MenuPanel.java
Feature to be tested	An Admin or Root user can delete an employee/staff account.
Feature Pass/Fail Criteria	Without losing any user information other than the one desired, an Admin or Root user should be able to find and delete a user by using their email address.
Means of control	Execute the application, Log-in as an Admin or Root then begin the Test Procedure. When the procedure is completed, the Test is over.
Data / Results	11/4/2016 9:29pm **PASS** All form validations and existing account checks are successful.
Test Procedure	With the application running, click on “Delete Staff Account”. A text box for an email address will be displayed. If the email address entered matches an email belonging to an employee-level account, it will be deleted upon clicking “Remove”. If that email account belongs to an account with access above employee, a message will display to notify the user the account will not be deleted. If the email is not found in the credentials file, a message with that information will be displayed. If “Success” is displayed, the account has been deleted and should no longer be in the credentials file.
Rationale	Functional Requirement # 33
Special requirements	If the credentials file is empty, parsing exceptions may occur causing the process to fail.

Test-case identifier	**DeleteAdminAccount_uc16**
Test location	./src/Model/EmployeeAccessModel.java ./src/View/EmployeeAccessView.java ./src/Controller/EmployeeAccessController.java ./src/Listeners/MenuActionListener.java ./src/GUIobjects/MenuPanel.java
Feature to be tested	A Root user can delete an Admin account.
Feature Pass/Fail Criteria	Without losing any user information other than the one desired, a Root user should be able to find and delete any user by using their email address.
Means of control	Execute the application, Log-in as Root then begin the Test Procedure. When the procedure is completed, the Test is over.
Data / Results	11/4/2016 9:33pm **PASS** All form validations and existing account checks are successful.
Test Procedure	With the application running, click on “Delete Administrator Account”. A text box for an email address will be displayed. If the email address entered matches an email belonging to any (non-Root) account, it will be deleted upon clicking “Remove”. If the email is not found in the credentials file, a message with that information will be displayed. If “Success” is displayed, the account has been deleted and should no longer be in the credentials file.
Rationale	Functional Requirement # 34
Special requirements	If the credentials file is empty, parsing exceptions may occur causing the process to fail.

Test-case identifier	**ProcessSale_uc21**
Test location	./src/Model/PointOfSaleModel.java ./src/View/PointOfSaleView.java ./src/Controller/PointOfSaleController.java
Feature to be tested	An registered user can sell a number of books to a customer.
Feature Pass/Fail Criteria	The user should be able to search for books by ISBN, add them to a queue of items to be purchased, then process the sale using the form of payments accepted by the customer: cash and credit.
Means of control	Execute the application, Log-in as Staff or higher then begin the Test Procedure. When the procedure is completed, the Test is over.
Data / Results	11/4/2016 11:10pm **FAIL** The sale would only process a single book – the last book added to the queue. Fixed: A data structure's contents was not being maintained while being passed between classes. 11/4/2016 11:40pm **FAIL** The subtotal was incorrect on the payment screen. Fixed: Floating point errors. 11/5/2016 12:11am **PASS** The books, total, order number, and sale file are reflecting matching information. Record of the sale is maintained.
Test Procedure	With the application running, click on "New Sale". Enter an ISBN into the search bar. Invalid responses will display a descriptive message. A valid ISBN will show the details of the corresponding book. Clicking on "Add" will put that item into the queue. When all items are added, press "Proceed to Payment". Enter the valid payment information and press "Process Payment". When the Receipt page is shown, the sale should have occurred and the sales file should have the details of the sale stored in it.
Rationale	Functional Requirement # 43
Special requirements	The inventory must have items that correspond to the entered ISBNs in order to be considered a "valid" ISBN.

Test-case identifier	**ProcessRefund_uc24**
Test location	./src/Model/PointOfSaleModel.java ./src/View/PointOfSaleView.java ./src/Controller/PointOfSaleController.java
Feature to be tested	An Admin user can refund an entire order to the customer.
Feature Pass/Fail Criteria	The user should be able to search for prior sales with the Sales ID. When a sale is found, the user should be able to click the corresponding button and remove the sale entry from the database.
Means of control	Execute the application, Log-in as Staff or higher then begin the Test Procedure. When the procedure is completed, the Test is over.
Data / Results	11/4/2016 11:49pm **FAIL** Inputting an invalid Sales ID would crash the application. Nullpointer exception. Fixed: Test for equality to null and handled as needed. 11/5/2016 12:05am **PASS** A vaild Sales ID immediately removes the sale from the sales file.
Test Procedure	With the application running, click on “Issue Refund”. Enter an Order ID into the search bar. Invalid responses will display a descriptive message. A valid Order ID will activate a previously-inactive button. Clicking on this button will prompt the refund. The sale should immediately be removed from the sales file.
Rationale	Functional Requirement # 63
Special requirements	The sales file should not be empty to avoid a possible parsing error.

Test-case identifier	**SearchForBookByTitle_uc1**
Test location	./src/Model/SearchForBooksModel.java ./src/View/ SearchForBooksView.java ./src/Controller/ SearchForBooksController.java
Feature to be tested	Search for a book by its title and have it appear in the search results
Feature Pass/Fail Criteria	The user should see the book's image, title, and ISBN properly formatted in the result display area of the interface.
Means of control	Execute the application, log in as any user, and begin the Test Procedure. When the procedure is completed, the Test is over.
Data / Results	11/9/2016 9:40am **FAIL** The search will always yield the desired book if the string input is part or all of the book's title *exactly*. If the search contains unnecessary white-space, such as "The Wizard of Oz", it would not find a case-insensitive match. 11/9/2016 9:50am **PASS** The search will now eliminate redundant whitespace from inside strings.
Test Procedure	With the application running, the user inputs search terms into the search bar and presses the Search button. Within a few seconds, the desired book should be shown in the results display.
Rationale	Functional Requirement # 1
Special requirements	None.

Test-case identifier	**SearchForBookWithCustomOptions_uc2**
Test location	./src/Model/SearchForBooksModel.java ./src/View/ SearchForBooksView.java ./src/Controller/ SearchForBooksController.java
Feature to be tested	Search for a book by its details and have it appear in the search results
Feature Pass/Fail Criteria	The user should see the book's image, title, and ISBN properly formatted in the result display area of the interface.
Means of control	Execute the application, log in as any user, and begin the Test Procedure. When the procedure is completed, the Test is over.
Data / Results	11/9/2016 10:21am **PASS** The search will always yield titles that contain the searched string.
Test Procedure	With the application running, the user inputs search terms into the search bar and presses the Search button. If that string is within the title, genre, or summary of the book, it should be part of the result set displayed in in the display section.
Rationale	Functional Requirement # 2
Special requirements	The string is not broken up into separate terms. "dog mouse cat" will find books that contain the whole string, not books that contain one or some of the words in the string.

Test-case identifier	**BrowseBooksByCategory_uc5**
Test location	./src/Model/SearchForBooksModel.java ./src/View/ SearchForBooksView.java ./src/Controller/ SearchForBooksController.java ./src/GUIObjects/CMPanel.java
Feature to be tested	Find books by using a built-in genre search
Feature Pass/Fail Criteria	The user should see the book's image, title, and ISBN properly formatted in the result display area of the interface if it has the genre selected.
Means of control	Execute the application, log in as any user, and begin the Test Procedure. When the procedure is completed, the Test is over.
Data / Results	11/9/2016 10:59am **PASS**. The genre search will yield non-duplicated result sets containing every book that is of the specified genre.
Test Procedure	With the application running, the user clicks on any of the buttons on the left side of the display under "Popular Genres:". The display should change to show books that are of the last genre clicked.
Rationale	Functional Requirement # 5
Special requirements	None.

Test-case identifier	**SearchOrderById_uc6**
Test location	./src/Model/SearchForOrderModel.java ./src/View/ SearchForOrderView.java ./src/Controller/ SearchForOrderController.java
Feature to be tested	Find orders by searching the database by the order number
Feature Pass/Fail Criteria	The user should see the order's contents: the books ordered and the quantity of each book ordered.
Means of control	Execute the application, log in as any registered user, and begin the Test Procedure. When the procedure is completed, the Test is over.
Data / Results	11/13/2016 4:40pm **FAIL** The display clears and no results are displayed. The formatting of the display was not correct. 11/13/2016 4:50pm **PASS** All matching order id's are displayed. A second click will allow the user to view the order.
Test Procedure	With the application running, the user clicks on the Orders menu option and the Accept Order Delivery button. In the text box that is displayed, the user enters the order id, exactly, and presses "Search by ID". If the order number entered is correct, it will be displayed within a few seconds.
Rationale	Functional Requirement # 6
Special requirements	None.

Test-case identifier	**AddWholeOrderToInventory_uc7**
Test location	./src/Model/AddOrderToInventoryModel.java ./src/View/AddOrderToInventory View.java ./src/Controller/AddOrderToInventory Controller.java
Feature to be tested	Accept order deliveries and update the inventory accordingly
Feature Pass/Fail Criteria	The user should be able to find any order, accept it, and see the inventory update appropriately.
Means of control	Execute the application, log in as any registered user, and begin the Test Procedure. When the procedure is completed, the Test is over.
Data / Results	11/13/2016 6:15pm **FAIL** The inventory was being updated incorrectly. The quantity was being subtracted instead of added. Changing the parameter to negative and using the methods to reduce inventory fixed the problem. 11/13/2016 6:30pm **PASS**.
Test Procedure	With the application running, the user clicks on the Orders menu option and the Accept Order Delivery button. The user can find any order in the system with either a date search or order-id search. When the result set is displayed, the user clicks on the appropriate order. The user will then see the books and quantity of books on the order. Leaving quantities unchanged, the user presses “Mark Received” to add the order to inventory.
Rationale	Functional Requirement # 7
Special requirements	If an order is green in the search result set, that order has already been accepted and cannot be accepted again.

Test-case identifier	**AddPartialOrderToInventory_uc8**
Test location	./src/Model/AddOrderToInventoryModel.java ./src/View/AddOrderToInventory View.java ./src/Controller/AddOrderToInventory Controller.java
Feature to be tested	Accept order deliveries and update the inventory accordingly
Feature Pass/Fail Criteria	The user should be able to find any order, accept it, and see the inventory update appropriately.
Means of control	Execute the application, log in as any registered user, and begin the Test Procedure. When the procedure is completed, the Test is over.
Data / Results	11/13/2016 6:30pm **PASS**.
Test Procedure	With the application running, the user clicks on the Orders menu option and the Accept Order Delivery button. The user can find any order in the system with either a date search or order-id search. When the result set is displayed, the user clicks on the appropriate order. The user will then see the books and quantity of books on the order. The user can change any books quantity with the number inputs and then press “Mark Received” to add the order to inventory.
Rationale	Functional Requirement # 7
Special requirements	If an order is green in the search result set, that order has already been accepted and cannot be accepted again.

Test-case identifier	**ChangeBookStockQuantity_uc11**
Test location	./src/Model/ModifyBookDetailsModel.java ./src/View/ModifyBookDetailsView.java ./src/Controller/ModifyBookDetailsController.java
Feature to be tested	See a book as it is listed in inventory and change its quantity in stock
Feature Pass/Fail Criteria	The user should be able to find any book, change its stock amount, and see that change reflected in the inventory immediately.
Means of control	Execute the application, log in as an Admin user, and begin the Test Procedure. When the procedure is completed, the Test is over.
Data / Results	11/14/2016 9:45am **PASS** Changes made to fix **ModifyBookDetails_uc10** automatically fixed this test-case.
Test Procedure	With the application running, the user clicks on the Inventory menu option then selects “Modify a book details/qty”. The user inputs the book’s ISBN and presses “Search”. When the book is displayed any number of options can be changed, including the stock level. Change the level as needed and save the changes. The inventory should be immediately updated.
Rationale	Functional Requirement # 11
Special requirements	Negative quantities will not be saved. They will be saved as zero.

Test-case identifier	**LogIntoSystem_uc17**
Test location	./src/Model/LoginModel.java ./src/View/LoginView.java ./src/Controller/LoginController.java ./src/Listeners/MenuActionListener.java ./src/login/credentials
Feature to be tested	A registered user can log into the application
Feature Pass/Fail Criteria	When the user logs in, the menu items should change to show the options relevant for that particular type of user.
Means of control	Execute the application, try to log in.
Data / Results	11/10/2016 6:11pm **FAIL** The log-in proving to be a poor place to change the menu display option. Need to find a way to change the menu in a similar way to how it's done on boot-up. 11/10/2016 7:02pm **FAIL** The User enum is not useful. Should use an integer instead 11/10/2016 7:31pm **FAIL** The menu is being redrawn on top of the existing menu instead of over it. 11/10/2016 7:39pm **FAIL** The CM icon is being pushed off the view after logging in. 11/10/2016 7:56pm **FAIL** When other options are checked after logging in, the menu changes back to an anonymous user's settings. 11/10/2016 8:20pm **PASS** A static variable to maintain the current user is kept to update the user whenever new options are selected. User can log in and stay logged in as intended.
Test Procedure	With the application running, the user clicks on the Log In button on the menu bar. The user fills in their username and password in the form and press submit. If the username and password are correct, the menu bar should change to show the appropriate menu items.
Rationale	Functional Requirement # 17
Special requirements	None.

Test-case identifier	**PlaceNewOrder_uc23**
Test location	./src/Model/PlaceNewOrderModel.java ./src/View/PlaceNewOrderView.java ./src/Controller/PlaceNewOrderController.java
Feature to be tested	An administrator or higher can create a new order
Feature Pass/Fail Criteria	When the user tries to create a new order, the orders file should show the order as created and it should be immediately accessible by other aspects of the program that monitor, change, or view orders.
Means of control	The user logs in as an Administrator and begins the test procedure.
Data / Results	11/19/2016 9:50am **FAIL** Invalid ISBNs in the middle of the form crashed the process and gave insufficient feedback to the user. Fixed to display which ISBN is actually invalid. 11/19/2016 11:30am **PASS** The order is not written until every book is validated.
Test Procedure	With the application running and logged in as an Admin user, the user clicks on the Orders menu option and selects “New Order”. The user inputs ISBNs and quantities until the order contains all the books desired. Then the user presses Place Order. The order should be immediately available in the allOrders file of the project.
Rationale	Functional Requirement # 23
Special requirements	The first item of the order must be filled out. Other sections can have blank ISBNs and they will be ignored.

Test-case identifier	**GenerateProfitLossReport_uc22**
Test location	./src/Model/GenerateReportModel.java ./src/View/GenerateReportView.java ./src/Controller/GenerateReportController.java
Feature to be tested	An administrator or higher generate a report showing sales and estimated profit over a time period.
Feature Pass/Fail Criteria	The user should be able to select a date range, see a formatted report of sales, and get a rough tally of profit based on cost and revenue.
Means of control	The user logs in as an Administrator and begins the test procedure.
Data / Results	11/20/2016 10:05am **FAIL** The varying amount of data makes it very difficult to format the output in a readable way. The content cuts off and is not displayed. 11/20/2016 10:51am **FAIL** The book titles, order numbers and dates are so long that they consume all the display space. Fixed by restricting their outputs to maximum lengths. 11/20/2016 11:30am **FAIL** The output is too difficult to read. Needs a table format. 11/20/2016 4:10pm **FAIL** The JTable is absolutely not right for this. Need to find an HTML method. 11/20/2016 6:32m **FAIL** The text is too small and un-readable. The text flows off the bottom of the application and is unreachable. Need a scroll pane and a way to adjust text size. 11/20/2016 7:01pm **FAIL** The scroll pane is working, but the size is still unacceptable. 11/20/2016 8.01am **PASS** A unique set of options allows the text to be sized in HTML format. Data is parsing and displaying appropriately.
Test Procedure	With the application running and logged in as an Admin user, the user Reports menu option and selects “Generate Sales Report”. The user picks a date range (default is the previous month + the current month) and submits. The display will provide a report that ends with estimated profit.
Rationale	Functional Requirement # 22
Special requirements	The user cannot select days. Only full months are options.

SECTION 5: GLOSSARY

Analysis Document	The part of the Requirements Analysis Document that contains the Use Cases, Scenarios, Analysis Object Model, Dynamic Model and the User Interface.
Administrator	The user that has access to all of the features that the root-administrator has except the ability to delete other administrators.
Analysis Object Model	Displays the different Class Diagrams to be used within the application.
Anonymous User	A user that is not logged into the Bookstore Desktop Application. The term can reference a customer or an employee that's not logged in.
Application	The Bookstore program designed for the customer.
Bookstore	A physical brick and mortar retail space that sells books. This does not reference an online e-commerce bookstore.
Class Diagram	A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia: https://en.wikipedia.org/wiki/Class_diagram).
COCOMO	The Constructive Cost Model (COCOMO) is a procedural software cost estimation model that uses a basic regression formula with parameters that are derived from historical project data and current as well as future project characteristics (Wikipedia: https://en.wikipedia.org/wiki/COCOMO).
Code Monsters	The team that's developing the Bookstore software.
Database	A storage container that contains data accessible by the Bookstore Desktop Application.
Database table	A table is a set of data elements (values) using a model of vertical columns (identifiable by name) and horizontal rows, the cell being the unit where a row and column intersect (Google Definition).
Dynamic Model	The dynamic model displays the visual representations of the interactions between categories that are required when implementing a Use Case.
Enterprise Level License	A license distributed to the customer allowing them to install as many copies as they choose to, and maintain the program through a third-party company if they choose to do so. The code and the program will be the property of the customer.
Expense Tracking System	The system that tracks business expenses such as gas, electricity, water, internet, etc.
Function Point Cost Analysis	Measures software size based on the functionality requested by the end user (http://www.washingtoniceaa.com/ Presentations/12_SW_Estimation_Function_Point.pdf).

Interaction Diagram	A subset of behavior diagrams that emphasize the flow of control and data among the things in the system being modeled (Wikipedia: https://en.wikipedia.org/wiki/Unified_Modeling_Language).
Interface	The Bookstore Desktop Application allowing for the interaction with the program through visual icons and other visual indicators.
Interface Diagram	The graphical representation of the Software Application to be built.
Inventory Management System	The system that tracks the inventory the Bookstore may have. The inventory management system only includes information relevant to the book, not the quantity in stock.
ISBN	The ISBN stands for the International Standard Book Number and is a unique book identifier.
Log File	A file that's kept on the server that tracks certain changes executed by users of the system.
Model/View/Controller	(MVC) is a software design pattern for implementing user interfaces on computers. A model stores data that is retrieved according to commands from the controller and displayed in the view. A view generates new output to the user based on changes in the model. It sends commands to its associated view to change the view's presentation of the model (Wikipedia: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller).
MySQL	MySQL is the world's most popular open source database (MySQL: https://www.mysql.com/about/).
Parallel Threads	A concept that allows the computer's CPU (the brain of the computer) to process multiple instructions concurrently.
Point of Sales (POS) System	System that allows for the acceptance of payment.
Primary Key	A primary key is a special relational database table column (or combination of columns) designated to uniquely identify all table records. A primary key's main features are: It must contain a unique value for each row of data. It cannot contain null values (Techopedia: https://www.techopedia.com/definition/5547/primary-key).
Registered User	A registered user is a Bookstore employee or owner.
Relational Database	A database structured to recognize relations among stored items of information (Google Definition).
Reporting System	A system that generates numerous reports that may include the profit-loss report, employee expense report and sales report.
Requirements Elicitation Document	Refers to this document and contains the collection of requirements of the system proposed by the customer.
Root Administrator	The main administrator using the software. The root administrator is the owner of the company in the case of the Bookstore Desktop Application.

Server	A computer that holds the data that the Bookstore Desktop Application needs to function. This data includes items such as the inventory and the users.
Software Architecture	Refers to the fundamental structures of a software system, the discipline of creating such structures, and the documentation of these structures. These structures are needed to reason about the software system. The architecture of a software system is a metaphor, analogous to the architecture of a building (Wikipedia: https://en.wikipedia.org/wiki/Software_architecture).
SQL	Stands for Structured Query Language and is a special-purpose programming language designed for managing data held in a relational database management system, RDBMS (Wikipedia: https://en.wikipedia.org/wiki/SQL).
Staff User	An employee that doesn't have administrative privileges.
System	Refers to the Bookstore Desktop Application.
System Exception	An exception that the system generates during an unpredictable software error.
System Log File	Stores system exceptions into a file accessible by the System Maintainer.
System Maintainer	The company that maintains the Bookstore software code.
Terminal	A terminal is a computer that runs the Bookstore Desktop Application. It can be a computer that the customer is using or the computer that the employee is accessing.
Test Case	A set of conditions under which a tester will determine whether an application, software system or one of the features is working as it was originally established for it to do. https://en.wikipedia.org/wiki/Test_case).
Use Case	A list of actions or event steps, typically defining the interactions between a role (actor) and a system, to achieve a goal. The actor can be a human or other external system (Wikipedia: https://en.wikipedia.org/wiki/Use_case).

SECTION 6: REVISIONS

Description	Date Changed
Merged WD document into this document. The WD document makes up Section 1 and each other section is incremented by one.	09/29/2016
RTM moved to start on a new page.	09/29/2016
Glossary moved to start on a new page.	09/29/2016
Added Use Cases to Section 4.7.1	09/30/2016
Added Interaction Diagrams to Section 4.7.2	09/30/2016
Added User Interface Diagrams to Section 4.7.3	09/30/2016
Updated User Interface Diagrams in Section 4.7.3	10/20/2016
Added Analysis Object Model to Section 4.7.4	10/20/2016
Added Dynamic Model to Section 4.7.5	10/21/2016
Updated Glossary with new terminology	10/21/2016
Updated heading for 4.7.5	10/30/2016
Added Test Cases to Section 4.7.6	11/04/2016
Modified Section 1.2 Team Roles	11/23/2016
Modified Section 1.3 Project Dates	11/23/2016
Modified Section 4.7.4 Analysis Object Model	11/24/2016
Modified Section 4.7.6 Test Cases	11/24/2016
Modified Section 5 Glossary	11/24/2016