## Introduction:

This project consisted of three phases, through which the purpose was to create a Book Store application. The first phase aimed to construct a Use Case Diagram to provide a high-level summary of the program and its functions, through a visual diagram. The second phase expanded on these functions by creating a class diagram for the program. The class diagram was used to model all the objects and classes that will be used to create this program. The final phase of this project implemented the program, based on the diagrams from the previous phases.

## Use Case Description:

A use case is used to represent all of the functionality provided by the program. The textual use case description consists of 7 different parts: Unique name, Participating actors, Flow of events, Entry conditions, Exit conditions, Exceptions, and Special requirements.

1. **Unique name:**
   - BookStoreApplication
2. **Participating actors:**
   - Owner and Customer
3. **Flow of events:**
   - As the owner, books can be managed after logging in. To add a new book, the owner enters the title of the book and the price, then clicks the "Add" button. The owner can delete books by first selecting the book of their choice and then pressing the "Delete" button

   - As the owner, customers can be managed after logging in. To add a new customer, the owner must enter the customer's username, password and number of points. After filling in this information, the owner then clicks the "Add" button to add the new customer. The owner can also delete customers by first selecting the user that they wish to delete and then pressing "Delete"

   - After logging in, the customer can add the selected book to their cart by pressing the "Add to Cart" button. The customer can go to their cart by pressing the "Go to Cart" button

   - After going to the cart, the customer can view all the books which were added to the cart. The customer can choose to buy all the books in the cart by using cash or by redeeming their points

- After purchasing the books, the customer gets to view their updated total amount of points and their updated status. The user must logout from this screen

4. **Entry conditions:**
   - The use case begins with the actors logging into the software

5. **Exit conditions:**
   - The use case ends with the actors logging out of the software

6. **Exceptions:**
   - When the user inputs an invalid username/password/book name/price, there will be a pop-up which lets the user know. The user must acknowledge this pop-up before continuing to use the application. An exception will be thrown if this were to happen.

   - When the user attempts to go to their cart, if there aren't any books in the cart at the time of clicking the button, a separate window will appear indicating that the user's cart is currently empty and, therefore they cannot view it.

   - When the user attempts to pay for their book(s) via redeeming their points, if the user has an insufficient amount of points, a separate window will appear indicating that they do not have enough points to complete the purchase

   - When the user attempts to add a new customer to the list, if the user does not fill in all of the required fields to add a new customer, a separate window will appear indicating that they have not filled all of the required fields

   - When the user attempts to create a new customer with a name that already exists, a separate window will appear indicating to the user that they cannot use this name as it already exists

7. **Special requirements:**
   - Assume that there is only one copy of each book

   - Book is deleted from the table once the book is purchased

## Rationale behind using the State Design Pattern:

The state pattern is used through the different scenes of the program in order to regulate the options based on who is currently using the program. This can be seen through the transition between the different screens such as the login and owner main screen. Furthermore, for each owner and customer we can individually regulate the different options of what the user can see depending on if the owner or customer is accessing the program. Thus, if only the owner is accessing the program then they can view/edit the books and customer submenus. If only the customer is accessing the program, the customer can access the books submenus and can buy books using cash or by redeeming the points. The state design pattern is also used to represent the different states the customer's account can be in based on their points. Depending on the number of points the customer currently holds, the customer's account status cycles between Silver and Gold status.