Overview:

This project report contains the Use Case Diagrams as well as the rationale behind the design of the final Project for COE528. In this report, it explains each of the Use Cases in great detail as well as explains the process that went into creating the Class Diagrams.

The goal of the project was to implement an interactive Book Store using object oriented analysis and design principles. Customers could browse and purchase books while also earning points to save on their next transaction.

This document outlines the key design decisions, emphasizing the use of design patterns and a structured class organization to create a functional bookstore application.

Use Case(s) Explained:

Use Case Name	Buy a Book
Participating Actors	Customer visits the online bookstore
Entry Conditions	This use case extends the “View all Available books”. It initialises once the customer logs in and accesses all available books.
Flow of Events	1. Customer logs into their account 2. Customer clicks buy a book a. For every dollar worth of the book item, 10 points are added to the Customer’s Status 3. After transaction is completed, the customer will be shown the total cost of the books as well as how many points they have
Exit Conditions	This use case terminates when the Customer exits “views all book” and then Customer logs out.
Exceptions	N/A
Special Requirements	N/A

Use Case Name	Manage Books
Participating Actors	Owner visits the bookstore (must be admin)
Entry Conditions	This use case includes “add books” and “remove books”. It initialises when the owner logins

Flow of Events	1. Owner logins into the Bookstore a. If invalid information, then error displays:“Username and password do not match” 2. Owner accesses Manage Books 3. Owner can add books in the form (name,price). a. In the case the owner wants to delete a book, the owner can select the book to delete and once deleted, the book will no longer appear in the list.
Exit Conditions	This use case terminates once the Owner logs out.
Exceptions	If a book is already added, a message will display: “Book already exists!” Another error that can occur is that the name of a book is an empty value and the price is 0 or invalid. Then an error message will occur:”Name can’t be empty/Price cannot be 0” The last error that can occur is when adding the book. If the system has a runtime error, the message will occur while saving the book; “Error saving books: (followed by the error message)”.
Special Requirements	Books must first be added to the book database by the owner.

Use Case Name	ManageCustomers
Participating Actors	Owner visits the book store (must be admin)
Entry Conditions	This use case includes “add customers’ and “remove customers”. It initialises once the owner logins.
Flow of Events	1. Owner Logs into the Bookstore a. If invalid information is provided, an error message is displayed: “Username and password do not match” 2. Owner accesses Manage Customers a. Owner adds customers to the list by inputting the username and password of the customer. b. Owner can also remove a customer by clicking on their name and removing it.
Exit Conditions	This use case terminates once the Owner logs out.
Exceptions	If a customer is already added (same username), then an error message is displayed “customer is already added”. If the file containing the customer list cannot be loaded, an error

	will display “File cannot be read.”.
Special Requirements	N/A

Use Case Name	Login
Participating Actors	Customer and Owner login to the bookstore
Entry Conditions	The valid login credentials for the Owner are username and password “admin”. The login credentials for a customer are added by the owner.
Flow of Events	1. The user enters the login page 2. The user enters their valid login credentials a. If the user is an owner, they are directed to the admin page b. Else if the user is a customer, they are directed to the customer start screen
Exit Conditions	The user is successfully authenticated and redirected to the appropriate screen (depending on whether they are a customer or owner).
Exceptions	If the user enters the incorrect login credentials, they are required to retry until they are correct. Incorrect login credentials indicate the user is entering an account that does not exist.
Special Requirements	For a customer, they must be added to the customer database. This is done through the owner’s login page and then ManageCustomers.

Use Case Name	Logout
Participating Actors	Customer and Owner log out of the bookstore
Entry Conditions	The customer or owner must be logged into the bookstore and on an appropriate screen
Flow of Events	1. User must log into the BookStore 2. User clicks the logout button at any point 3. User returns to the login screen

Exit Conditions	User successfully logged out and returned to the login screen
Exceptions	The only way a user cannot log out if they are somehow not properly logged in.
Special Requirements	N/A

Use Case Name	Redeem Points & Buy a Book
Participating Actors	Customer accesses redeem points and buy a book
Entry Conditions	This use case extends the “View all books” case. It initialises once the customer logs into their account and accesses “redeem points and buy books”.
Flow of Events	1. Customer must login with their information a. If invalid information is provided, then error occurs 2. Customer chooses redeem points and buy a book 3. Customer purchases their book(s) with the price being reduced by one dollar for every 100 points that the Customer has. a. The Customer State is then kept as silver if the customer’s points are below 1000 points if , or changes to gold if it goes above.
Exit Conditions	Customer completes the purchase and is then directed to the main menu “view available books”.
Exceptions (what happens if things go wrong)	If invalid login information, then customer is given the error: “Username or password does not match”
Special Requirements	N/A. Customers can still have 0 or less than 100 points to access this class.

Use Case Name	View Available Books
Participating Actors	Customer accesses View available books after logging in
Entry Conditions	This use case starts once the Customer logs into their account.

Flow of Events	1. Customer must login to their account a. If invalid information is given, then an error message appears:“Username and password do not match” 2. Customer now has access to view available books a. Customer can choose to buy books or redeem points and buy books at this page b. Customers can also view their status (how many points they have or what is their current state)
Exit Conditions	This use case terminates once the Customer logs out of their account.
Exceptions	If the book list cannot be loaded, an error message will display:”Error Loading Books”
Special Requirements	N/A

Rationale behind the Class Diagram

This class diagram describes how we developed a simple bookstore application. Using a state design pattern, we implemented the basic functionality of the classes, such as user management, customer states, and book and user management.

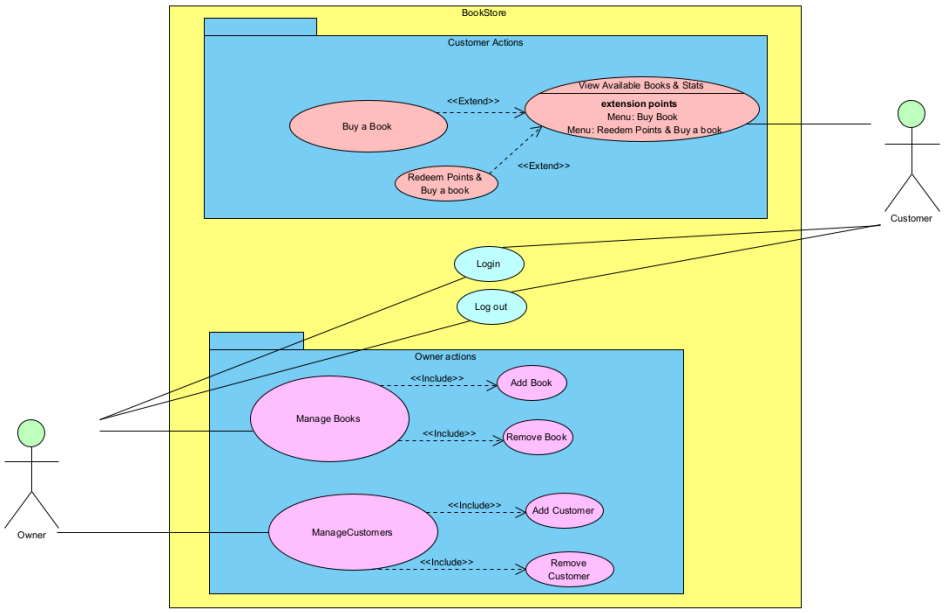
To implement user management, we made an abstract user class that extends the owner and customer classes. To ensure there is only one owner and a single book and user manager, we used a singleton design pattern.

To implement the customer states, we made a “CustomerState” interface that implements the two states, the golden state and the silver state. The state design pattern allows the customer class to change its behavior dynamically based on its state (GoldenState or SilverState). Instead of using if-else statements inside the Customer class, the pattern delegates behavior to state objects.

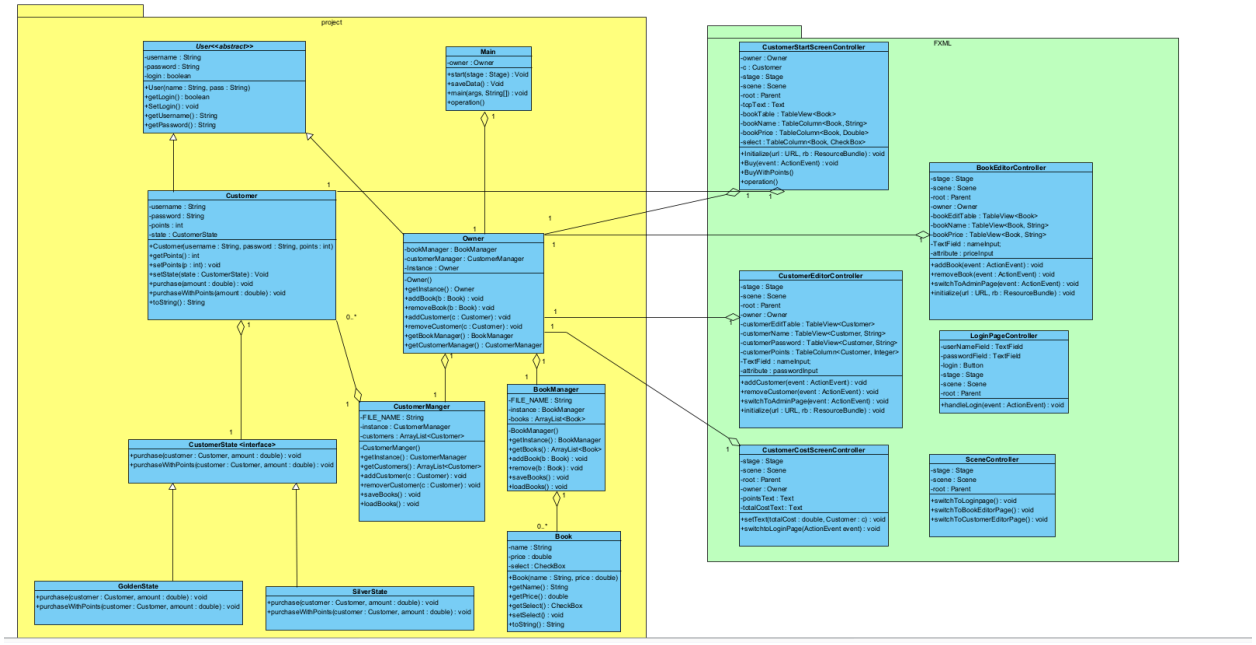
To implement the book and user management, we created “CustomerManager” and “BookManager” to create functions used to either store books, create books, remove books, and the same for customers. These classes also implement the idea that when a user clicks the [x] exit button, it saves the user's data in the files “customer.txt” and “books.txt”.

Lastly, we created classes such as “CustomerStartScreenController, CustomerEditorController, BookEditorController, LoginPageController, CustomerCostScreenController, and ScreenController” to separately handle the single-window GUI operations of our application.

Appendix



Use Case Diagram



Class Diagram