Report Phase 4

Student Name	Nathan Vu	Dylan Weston	Eric Ren	David Shi	Lama Mohamed
Student ID	501120037	500961447	501039074	501090024	501042394
Signatu re*	[illegible]	DV	[illegible]	DS	[illegible]

<u>Use Case Description:</u>

Name: LoginScreen
Participating Actors:

- Customer
- Owner

Flow of Events:

1. The login screen pops up with the message “Welcome to the BookStore App”
2. Two fields named “Password” and “Username” appear that can be filled in
3. A “Login” button appears at the bottom

Entry Condition: This use case starts when the program is run for the first time.
Exit Condition: This use case terminates when the login button is clicked with a registered username and password, or when the [x] button is clicked.
Exceptions:

- The user is notified immediately if the username or password entered is incorrect
- The user is notified immediately if either the username of password field is left blank

Special Requirements:

Name: OwnerStartScreen
Participating Actors:

- Owner

Flow of Events:

1. Three buttons labeled “Books”, “Customers”, and “Logout” are displayed

2. When owner clicks on “Books”, the ownerBookScreen is displayed
3. When owner clicks on “Customers”, the ownerCustomerScreen is displayed
4. When owner clicks on “Logout”, the owner is taken back to the login screen

Entry Condition: This use case starts when the owner logs in with a registered username and password.
Exit Condition: This use case terminates when the owner clicks on when of the three buttons on the screen.
Exceptions:
Special Requirements:

Name: OwnerBooksScreen
Participating Actors:
- Owner

Flow of Events:
1. The top of the screen contains a table with two columns “Book Name” and “Book Price”
2. A row in the table contains a name of a book and the price of the book
3. The middle part of the screen contains two text fields labeled “Name” and “Price”, along with a “Add” button
4. When owner enters the name of a book and its price then clicks the “Add” button, the name of the book along with the price is displayed on the table
5. When the owner clicks the “Delete” button at the bottom of the screen after selecting a row on the table, the row is deleted from the table
6. When the owner clicks the “Back” button at the bottom of the screen, the owner is taken back to the ownerStartScreen

Entry Condition: This use case starts when the owner clicks on the “Books” button from the ownerStartScreen.
Exit Condition: This use case terminates when the owner clicks on the “Back” button at the bottom of the screen.
Exceptions:
- The owner will be notified that “No rows is selected” when clicking the “Delete” button without selecting a row

Special Requirements:

Name: OwnerCustomersScreen
Participating Actors:
- Owner

Flow of Events:
1. The top of the screen contains a table with three columns labeled “Username”, “Password”, and “Points”
2. The middle of the screen contains two text fields labeled “Username” and “Password” along with a button “Add”
3. When the owner enters a customers username and password then clicks “Add”, a new row with the customers information appears on the table at the top
4. When the owner clicks the “Delete” button at the bottom of the screen after selecting a row on the table, the row is deleted from the table
5. When the owner clicks the “Back” button at the bottom of the screen, the owner is taken back to the ownerStartScreen

Entry Condition: This use case starts when the owner clicks on the “Customer” button from the ownerStartScreen.
Exit Condition: This use case terminates when the owner clicks on the “Back” button at the bottom of the screen.
Exceptions:
- The owner will be notified that “No rows is selected” when clicking the “Delete” button without selecting a row

Special Requirements:

Name: customersStartScreen
Participating Actors:
- Customer

Flow of Events:
1. A welcome message with the customers name and points is displayed
2. The customer’s status, either gold or silver is displayed
3. The customer's status is silver if the customer has less than 1000 points and the status is gold if the customer has 1000 points or above
4. The middle part of the screen contains a table with three columns labeled “Book Name”, “Book Price”, and “Select”
5. A row in the table contains the name of a book, the price of the book, and a checkbox that can use either checked or unchecked
6. The bottom part of the screen contains three buttons labeled “Buy”, “Redeem Points and Buy”, and “Logout”
7. If the customer clicks on “Buy” or “Redeem Points and Buy”, the customersCostScreen will be displayed
8. If the customer clicks on “Logout”, the customer will be taken back to the loginScreen

Entry Condition: This use case starts when the customer enters their registered username and password and clicks on the "Login" button on the LoginScreen.
Exit Condition: This use case terminates when the customer clicks on the "Logout" button at the bottom of the screen.
Exceptions:
Special Requirements:

Name: CustomerCostScreen
Participating Actors:
- Customer

Flow of Events:
1. The top of the screen is the message "Total Cost" which is the total transaction cost
2. The total cost of the books is the sum of the individual books without tax
3. The middle part of the screen displays the the points and status of the customer
4. When the owner clicks on the "Logout" button at the bottom of the screen, the customer is taken back to LoginScreen
5. If the customer clicked on "Redeem Points and Buy", the total transaction cost is calculated after all the points are redeemed
6. For every 1 CAD spent, the owner earns 10 points and for every 100 points redeemed, 1 CAD is deducted from the total cost

Entry Condition: This use case starts when the customer clicks either the "Buy" or "Redeem Points and Buy" button on the CustomerLoginScreen.
Exit Condition: This use case terminates when the customer clicks on the "Logout" button at the bottom of the screen.
Exceptions:
- The total transaction cost must not be less than 0

Special Requirements:

<u>Reasons for Using the State Design Pattern</u>

The state design pattern is implemented when an object's behavior depends on its state and it must change its behavior at run time depending on that state. It is also applicable when methods have large multipart conditional statements that depend on the object’s state. In the case of a bookstore application, the state pattern is compatible due to the varying state of the user. The user of the application can be either the owner, or multiple customers. The program functionality changes depending on whether it is a customer or owner logging in. For example if an owner logs in, then the program should allow the user to add/remove customers whereas if a customer logs in, the customer would not have the access to do so. The bookstore application also has parts where it has to display user information such as the customer status and points. In this case, depending on which customer logs into the application, the points and status would have to change to correspond to the customer. A new customer that has 0 points would not be able to redeem points and buy which means the method for that functionality should be conditional depending on the customer. The classes that participate in the state pattern design include the loginScreen, ownerCustomersScreen, customerLoginScreen, and the customerCostScreen. Overall, it is clear that the state design pattern is best suited for this bookstore application due to the various conditions of the user logging into the system.