COE528 S2022 Final Project

Govinda Mazumder

500817562

Describe use-case:

The two actors shown in the use-case diagram are the manager and the customer. The entry condition is that both actors must provide their login credentials (which are compared with the existing database of user login and passwords) at the start of the program in order to successfully login and access the bookstore app. The difference in the actors comes from their clearance to access capabilities within the app. The event flow for the manager is as follows, after satisfying the entry condition the manager is able to add and delete entries into the databases for both customers and books to be sold. The event flow for the customer is after logging in, the customer is able to buy books and buy books using points, after selecting either option they receive a total cost and updated points balance and status. Points accumulated from the purchases are stored along with the user data, so that the customer can continuously purchase books with or without points and this would impact their status between Gold and Silver. Accumulated points are representative of status (upwards of 1000 points is Gold and anything less is silver) and can be used to decrease overall cost. The exit condition for both actors is to logout of the app when their actions are complete, which is a complete transaction for the customer and after registering a book or customer for the manager.

Describe rationale behind State Design Pattern:

The rationale for using the sate design program is related to the general function of the bookstore app. At any time either a customer or manager is performing operations on the application, which involves a different mode or state with different parameters. The ability to transition between these finite states expediently based on internal state fits the general requirement for using the state design pattern, which was originally based on the similar concept of finite state machines. As demonstrated in my code, there are several states with their respective controllers which act as a means of operating and transitioning between the states (i.e CustomerCheckoutScreen & CheckoutController). When prompted by the user either by entering information or pressing a button, the program is able to react appropriately and move to perform the necessary functions required to complete the task outlined by the project requirements. The states in the context of the project are the different screens and displays the customers and the manager see, each of which performs unique functions. All the screens are connected in someway, which allows for users to navigate between them as needed. This is the overall objective of the state design pattern, to move between states instantaneously by passing references based on the current internal state. Each state and state controller occupies a class that houses the state specific behaviour, which provides ease of access for modifying and updating the code as required.