Report

The use of a case diagram is a way to summarize details of a system and the users within that system. In our project a case diagram is a perfect way to be able to explain our project. The way I learned about the user-case is that there are seven parts. The seven parts include unique names, participating actors, flow of events, entry condition, exit condition, exceptions, and special requirements. The unique name for one of ours is “log in”. The participating actors in our project are the customer and owner. The entry condition is where the use case starts when the Owner and the Customer are logged into the system. The flow of events would start by the login and then start to both the Owner and Customer. The way the customer works would be by them selecting the books, and they have two ways to purchase which would be by cash or redeemed by points. These would have to include “very sufficient funds”. For the Owner it starts by viewing the books and the customers, they will both have the option to return to the previous screen. For books you will have the option to add and remove them, and for the customer you can remove and add them as well. Then they are all connected to the logout. The exit condition is where the use case terminates when the Owner and Customer are logged out of the system. The exceptions would be when the user inputs a wrong password or username. If this was the case there they would get an output of the screen saying it is invalid. Another one would be if there was an error during the runtime, or if the connection was lost. If this was to be the case then it would be an exception. The special requirements in our project would be after a customer buys a book, then the book should be deleted from the book table. That is the user-case of our project.

The state design pattern was used to handle situations where a class's or method's behaviour depends on its present state during the development of the bookstore application. This design pattern is appropriate for changes to runtime state that don't impact the present class or state. The User class in the application can be in either the Owner or Customer form, each with its own set of functionalities. Each state was represented by a different class in order to apply the state pattern. The User class is the "client," the GUI is the "context," and there are two states (Owner and Customer), each with a number of sub-states, according to the state machine model. There are three sub-states under the Owner state: Books, Customers, and Logout. The collection of books could be accessed by clicking on the Books sub-state, and the list of customers could be accessed by clicking on the Customers sub-state. The user would log out and be brought back to the login page by clicking on the Logout sub-state. Three additional states exist under the Customer state: Buy, Redeem, and Logout. The Buy sub-state, as well as the Redeem and Buy sub-states, would transport the user to another window showing the total cost and status, respectively, upon clicking on them. Finally, the user could log out of the program by selecting the Logout sub-state. That is the state design pattern for our project.