Component-level Design Review Report

In our Component-level design folder we have four files to show the functionality of our project at a component-level. The four files, “Sequence DIagram BorrowBook Final Project.png”, “Sequence DIagram Class Final Project.png”, “Sequence Diagram Modify Entry Final Project.png”, and “state machine diagram.png” refer to the Borrow Book SD, Class SD, Modify Entry SD, and State Machine Diagram respectively.

SD Borrow Book

The sequence diagram for our borrow book use case is most likely our most complex use case. The reason for this is that the use case actually incorporates every class in the project. This can be seen when the Home Controller grabs both the bookID and the CardNumber from their respective classes. After this, the Home Controller creates both a book and account object to be utilized in conjunction with the Borrowing class to borrow the book.

Within the sequence diagram there are a few key decisions that are made. The first is whether or not the selected book by the customer actually has enough copies so that it may be signed out. To do this, the Home Controller utilizes the Book class to grab the BookID and then internally checks the available copies. If the number of copies is less than 1 or equal to zero, a break statement is fulfilled and the sequence diagram terminates with no book being borrowed. The next decision is very similar, though the Home Controller is just verifying that the user has not signed out more than five books.

SD Class Structure

The sequence diagram for the class structure (class-level) is a sequence diagram that shows the simple functionality behind each major function of each class. This is done while showing the intricate way that the classes interact with each other and the database. One thing to note is that this sequence diagram does not include every function, as some are too simple to represent in this way.

SD Modify Entry

The sequence diagram for our modify entry use case is also one of our largest and most complex use cases, but this time for the Employee actor. The main functionality of this use case is to edit or change attributes in regards to a specific book.

Like the Borrow Book sequence diagram, the Modify Entry sequence diagram makes two key decisions. The first is when the Database must verify whether or not the specific book chosen by the Employee actually exists in the database, if the BookID is not found the sequence is terminated. The next key decision is when the attributes that the Employee wishes to change are validated. If the attributes inputted by the Employee are not validated for whatever reason (i.e. a negative number, profanity etc.) the sequence is terminated.

State Machine Diagram

The state machine diagram shows an overall sequence for moving through the project. For example, if a Customer would like to borrow a book, they first enter the state of being signed out, they register, sign in with account ID, search the book database, borrow a book, then sign out. Each of those specific instances represent a change of state within the Home Controller system. Some key identifying aspects of our state machine diagram include the Register() function, the book editor form, and the Book Search state. The Register() function is actually a self-call to the Signed out state, as a user will register with credentials, but not be automatically signed in, as they still have to enter correct credentials after registering. Furthermore, the Book Editor Form is unique as three separate functions all point to the same form, though each function interacts with the form differently, creating a single space for Employees to modify the database. Additionally, the last unique feature of our state machine diagram would be the Book Search state. Whats interesting about this state is that it actually encompasses much of the system as a whole. This is because even while you are viewing your account page, you are still actively searching through the database for books.