Team: Nuha Albadi
Mukund Madhusudan Atre
Karthik Handady

Title: Library Checkout System

1. List the features that were implemented (table with ID and title).
2. List the features were not implemented from Part 2 (table with ID and title).
3. Show your Part 2 class diagram and your final class diagram. What changed? Why? If it did not change much, then discuss how doing the design up front helped in the development.
4. Did you make use of any design patterns in the implementation of your final prototype? If so, how? Show the classes from your class diagram that implement each design pattern (each design pattern as a separate image in the .PDF). If not, where could you make use of design patterns in your system? Show a class diagram of how you could implement each design pattern and compare how it would change from your current class diagram.
5. What have you learned about the process of analysis and design now that you have stepped through the process to create, design and implement a system?

1. The features that were implemented

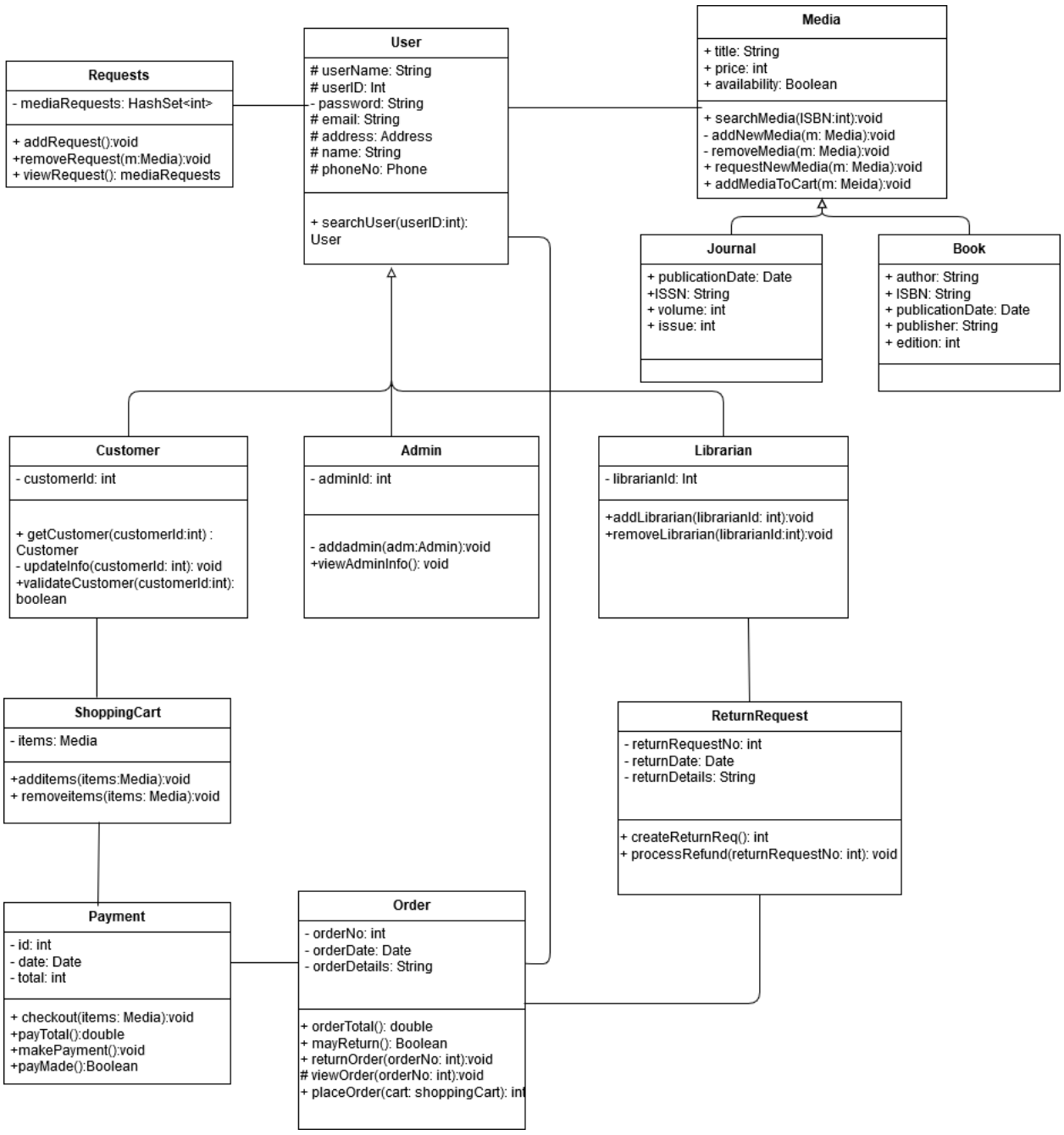
User Requirements	
ID	**Requirement**
UR-001	A customer must be able to sign up for the system.
UR-002	The admin must be able to add a librarian to the system.
UR-003	The Librarian and Admin must be able to alter a customer's information.
UR-004	The admin can add media
UR-006	The customer can search for media.
UR-007	The customer or librarian can file a request for new media.
UR-008	The admin and librarian can search for a customer's account
UR-009	The customer can add media to their shopping cart.
UR-010	The customer can checkout media in their shopping cart

2. The features that were not implemented

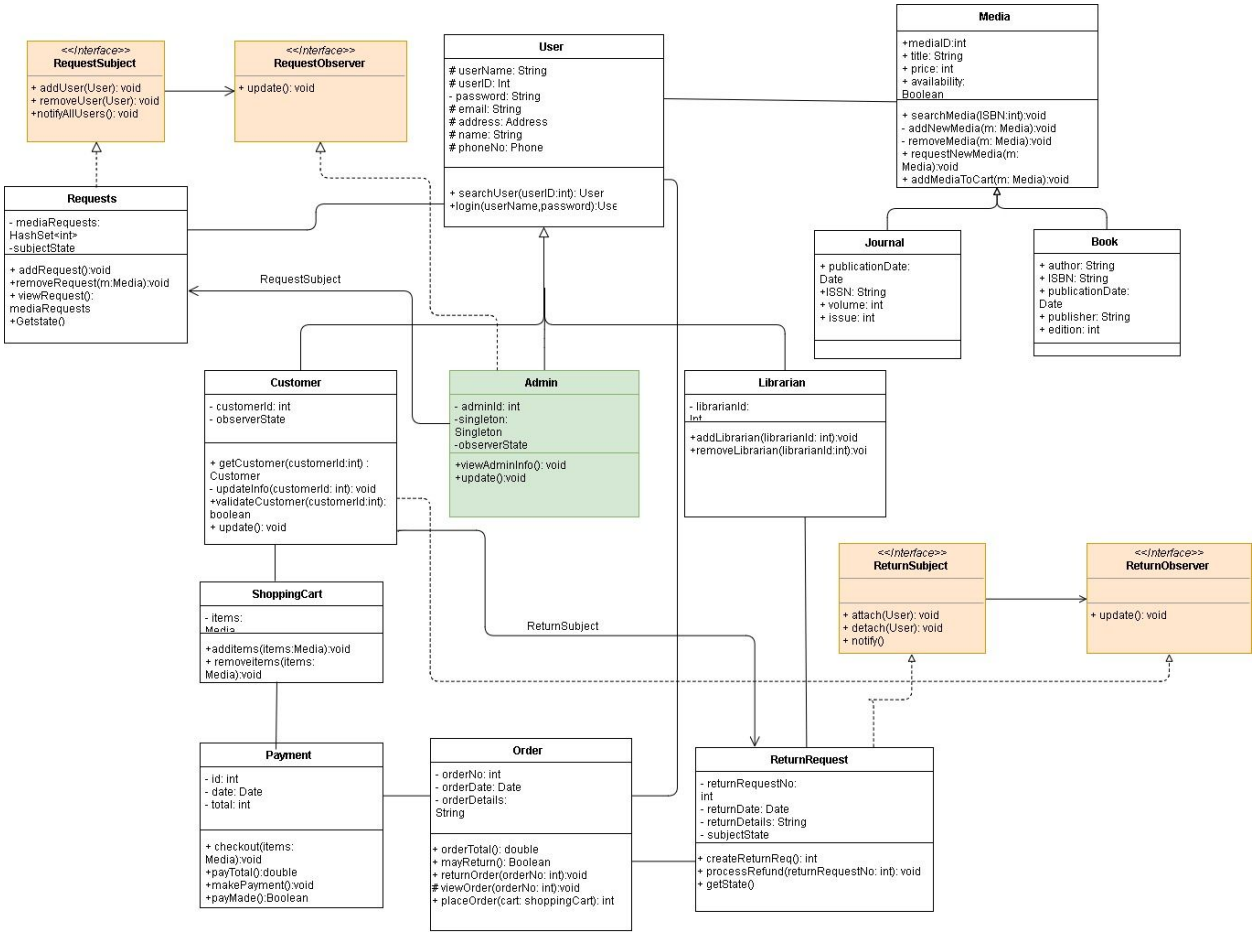
User Requirements	
ID	**Requirement**
UR-005	The admin can remove media
UR-011	The customer and librarian can view customer's order
UR-012	The customer can request a return
UR-013	The librarian can process a refund

3.

Part 2 class diagram:



Final class diagram:



Color coding scheme for two design patterns in the class diagram is shown below:
Peach color: Observer Design Pattern
Green color: Singleton Design Pattern

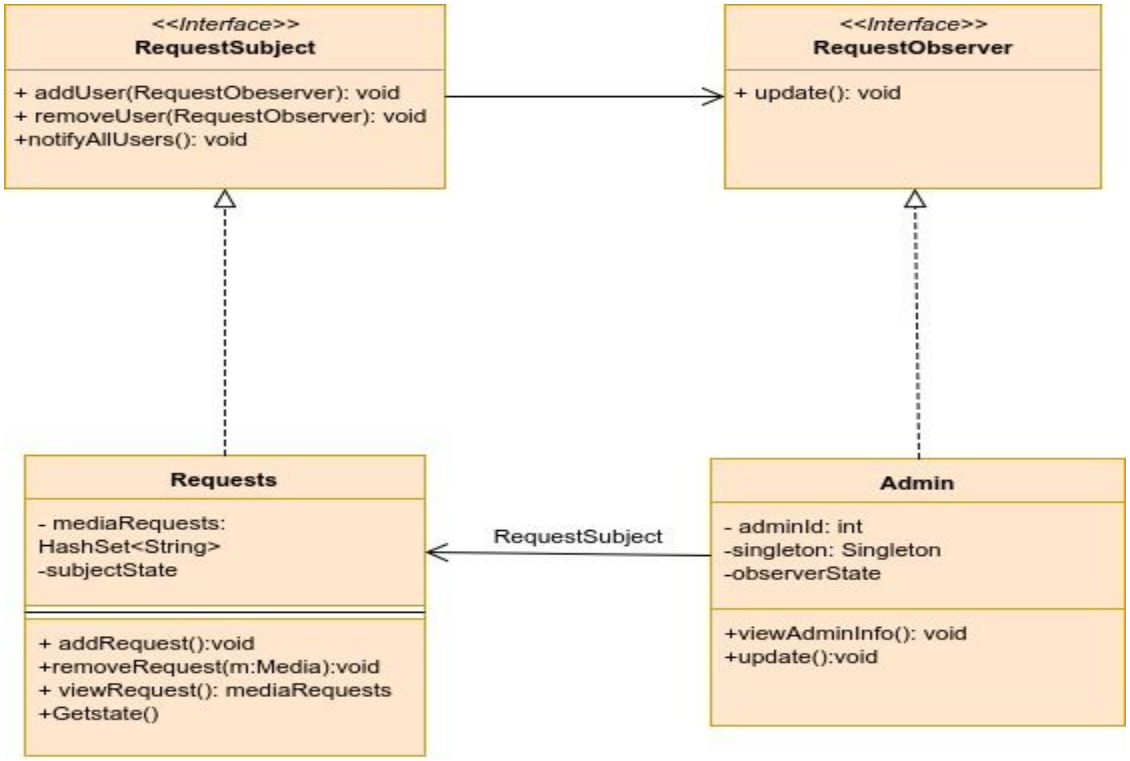
There were several changes that were done to the design and implementation of the project. The major changes were the implementation of the design patterns such as observer and singleton. Adding these two design patterns made the implementation of some use cases much simpler. The small changes were addition of variables and functions that became apparent while trying to implement the program. Planning the design upfront was also very helpful for organized thinking which is required for software design. We understood how the user requirements will be implemented in software beforehand and this saved us from the constant struggle of changing our implementation again and again when new user requirements came up. Also, with the help of UML diagrams every team member could convey their ideas in a common

representational form. We gained knowledge about the actual form of Design Patterns through the lectures and applied them during refactoring.

4.

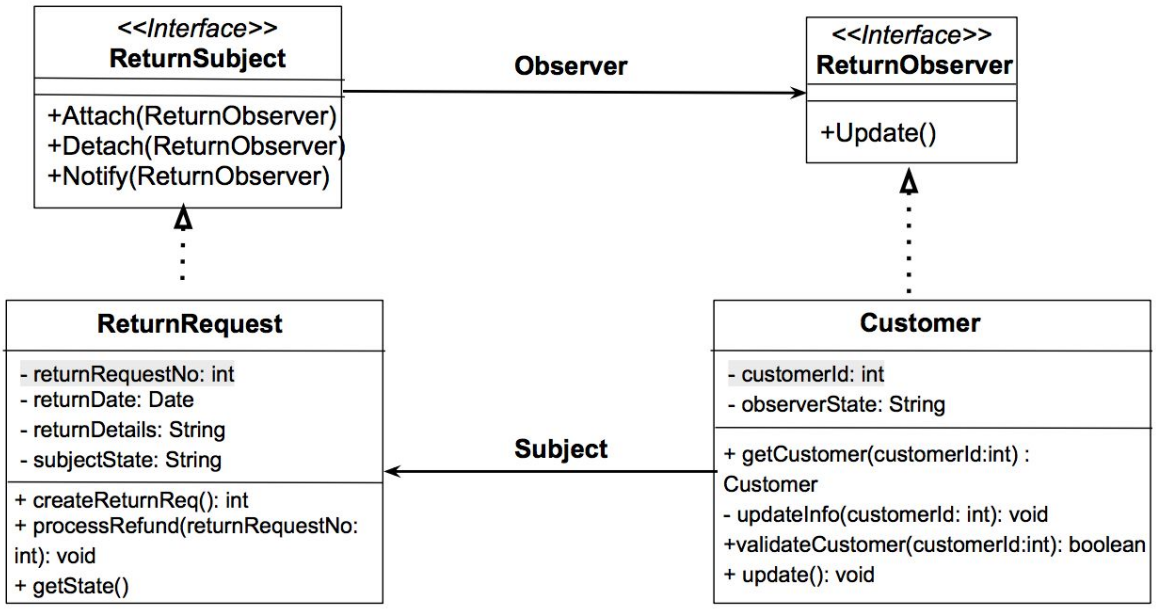
We implemented the Observer design pattern for new media requests and return requests in our project.The Observer design pattern helped us in setting up a notification system for relevant subscribers in our project in a sophisticated way. We also implemented the Singleton design pattern for the Admin class which enables us to keep only one instance of admin.

Observer design pattern for new media requests:



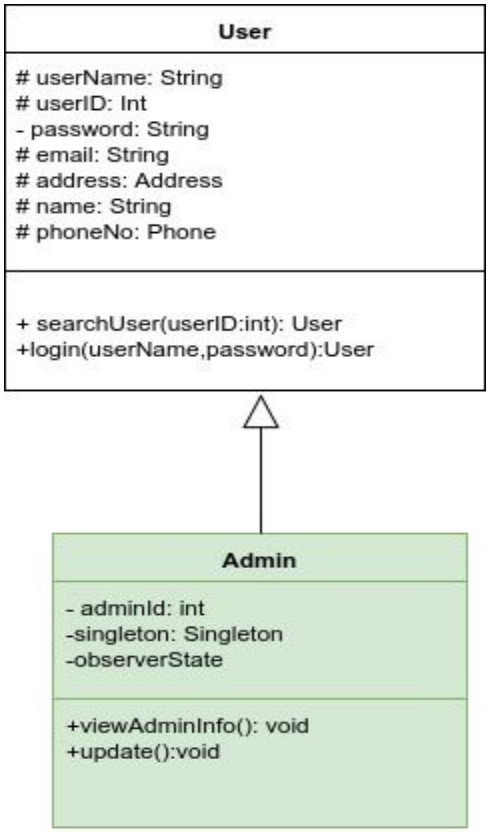
Requests is the ConcreteSubject class which stores state of new media requests and send a notification to observers (Admin) when it is changed. The admin should be notified of requests for adding new media to the library so that he can add media to the library.

Observer design pattern for return requests:



ReturnRequest is the *ConcreteSubject* class which stores state of return requests and send a notification to observers (Customer) when it is changed. This was a necessary addition as there was no way for the customer to know if the return was approved before implementing the pattern.

Singleton Design Pattern for Admin class:



5.

We have learned that it is extremely important to spend a good amount of time on analysing and designing the system before implementing it. Software is naturally intangible which makes it hard to convey meaning between use (the developers). However, the diagrams that we created enabled us to share a common understanding and made sure that we are all on the same page. The refactoring portion showed how the original design is usually flawed and many iterations are necessary to ensure a good design. The class diagram made us think thoroughly about the classes that we will have along with attributes and methods. So, most of the implementation part was creating those same exact classes. The design patterns that we learned during class helped us solve some design issues in our system and will surely help use with future design problems.