Team: Nuha Albadi
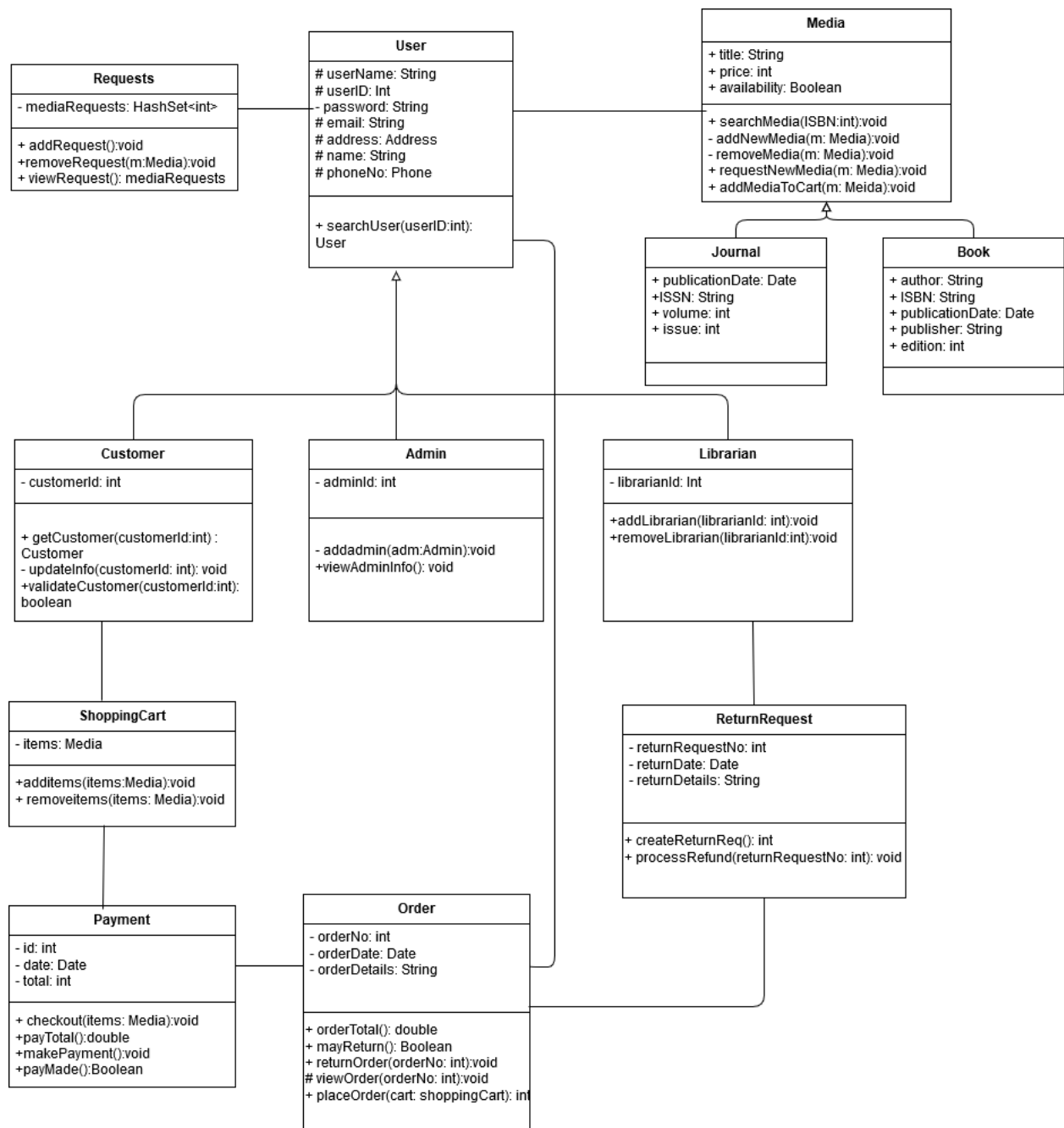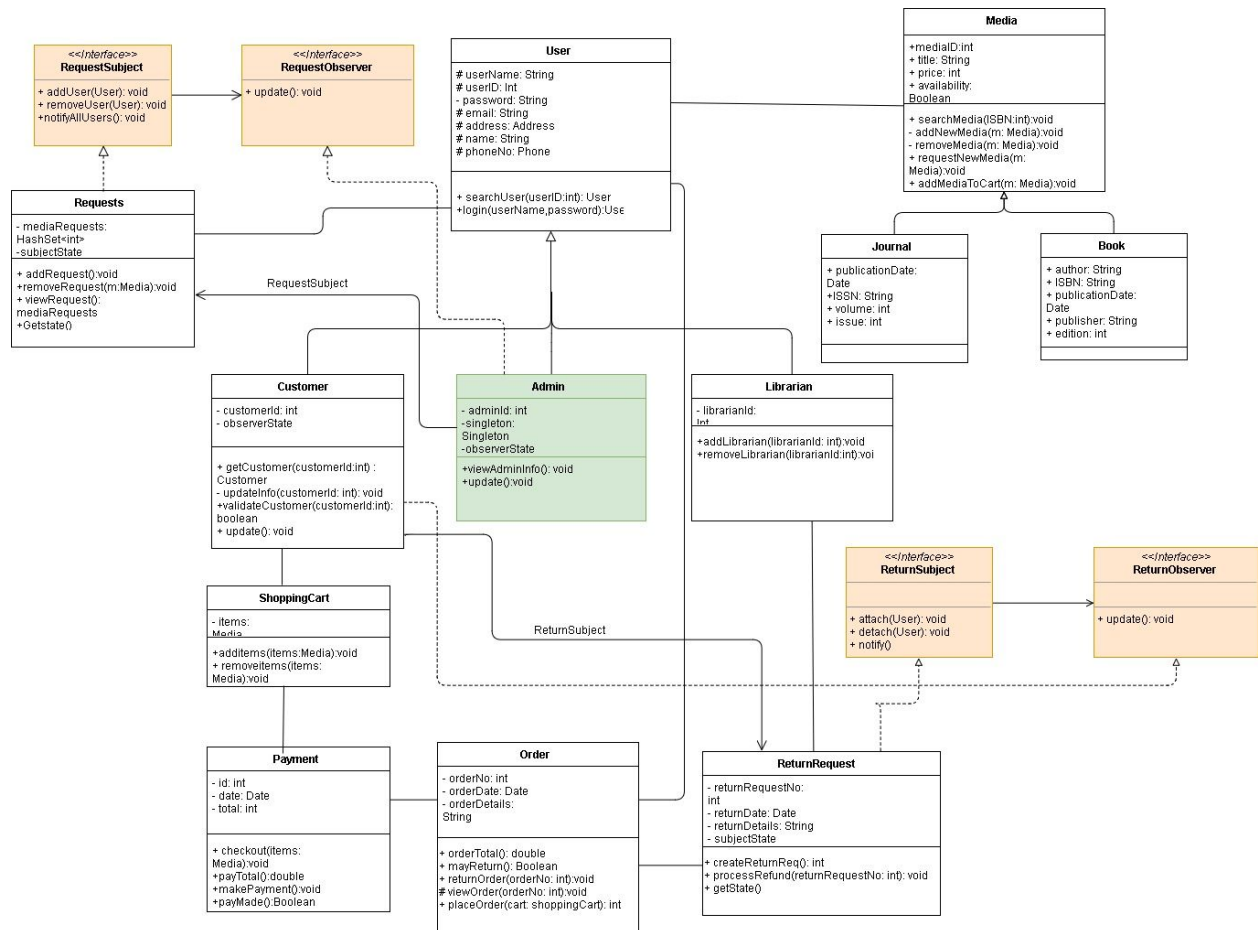   Mukund Madhusudan Atre
   Karthik Handady

Title: Library Checkout System

Project Part 2 Class Diagram:

Requests

- mediaRequests: HashSet<int>

+ addRequest():void
+removeRequest(m:Media):void
+ viewRequest(): mediaRequests

User

userName: String
userID: Int
- password: String
email: String
address: Address
name: String
phoneNo: Phone

+ searchUser(userID:int): User

Media

+ title: String
+ price: int
+ availability: Boolean

+ searchMedia(ISBN:int):void
+ addNewMedia(m: Media):void
- removeMedia(m: Media):void
+ requestNewMedia(m: Media):void
+ addMediaToCart(m: Meida):void

Journal

+ publicationDate: Date
+ISSN: String
+ volume: int
+ issue: int

Book

+ author: String
+ ISBN: String
+ publicationDate: Date
+ publisher: String
+ edition: int

Customer

- customerId: int

+ getCustomer(customerId:int) : Customer
- updateInfo(customerId: int): void
+validateCustomer(customerId:int): boolean

Admin

- adminId: int

- addadmin(adm:Admin):void
+viewAdminInfo(): void

Librarian

- librarianId: Int

+addLibrarian(librarianId: int):void
+removeLibrarian(librarianId:int):void

ShoppingCart

- items: Media

+additems(items:Media):void
+ removeitems(items: Media):void

ReturnRequest

- returnRequestNo: int
- returnDate: Date
- returnDetails: String

+ createReturnReq(): int
+ processRefund(returnRequestNo: int): void

Payment

- id: int
- date: Date
- total: int

+ checkout(items: Media):void
+payTotal():double
+makePayment():void
+payMade():Boolean

Order

- orderNo: int
- orderDate: Date
- orderDetails: String

+ orderTotal(): double
+ mayReturn(): Boolean
+ returnOrder(orderNo: int):void
# viewOrder(orderNo: int):void
+ placeOrder(cart: shoppingCart): int

New Class Diagram:

<<Interface>> RequestSubject
+ addUser(User): void
+ removeUser(User): void
+notifyAllUsers(): void

<<Interface>> RequestObserver
+ update(): void

User
userName: String
userID: Int
- password: String
email: String
address: Address
name: String
phoneNo: Phone

+ searchUser(userId:int): User
+login(userName,password):Use

Media
+medialD:int
+ title: String
+ price: int
+ availability: Boolean

+ searchMedia(ISBN:int):void
- addNewMedia(m: Media):void
- removeMedia(m: Media):void
+ requestNewMedia(m: Media):void
+ addMediaToCart(m: Media):void

Requests
- mediaRequests: HashSet<int>
-subjectState

+ addRequest():void
+removeRequest(m:Media):void
+ viewRequest(): mediaRequests
+Getstate()

RequestSubject

Journal
+ publicationDate: Date
+ISSN: String
+ volume: int
+ issue: int

Book
+ author: String
+ ISBN: String
+ publicationDate: Date
+ publisher: String
+ edition: int

Customer
- customerId: int
- observerState

+ getCustomer(customerId:int) : Customer
- updateInfo(customerId: int): void
+validateCustomer(customerId:int): boolean
+ update(): void

Admin
- adminId: int
-singleton: Singleton
-observerState

+viewAdminInfo(): void
+update():void

Librarian
- librarianId: Int

+addLibrarian(librarianId: int):void
+removeLibrarian(librarianId:int):voi

<<Interface>> ReturnSubject
+ attach(User): void
+ detach(User): void
+ notify()

<<Interface>> ReturnObserver
+ update(): void

ShoppingCart
- items: Media

+additems(items:Media):void
+ removeitems(items: Media):void

ReturnSubject

Payment
- id: int
- date: Date
- total: int

+ checkout(items: Media):void
+payTotal():double
+makePayment():void
+payMade():Boolean

Order
- orderNo: int
- orderDate: Date
- orderDetails: String

+ orderTotal(): double
+ mayReturn(): Boolean
+ returnOrder(orderNo: int):void
#viewOrder(orderNo: int):void
+ placeOrder(cart: shoppingCart): int

ReturnRequest
- returnRequestNo: int
- returnDate: Date
- returnDetails: String
- subjectState

+ createReturnReq(): int
+ processRefund(returnRequestNo: int): void
+ getState()

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

We applied the following design patterns:
1. Observer: The Observer design pattern is applied in two different use cases. 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 successful before implementing the pattern. The observer design pattern is also applied to Request New Media use case. This enables the system to notify the Admin and any other subscribed User(Librarian possibly) to get notified when a new Media is requested by the Customer. For now, only admin

will get notified of Requests for New Media. The code is open for extension in a way that Librarians can also be added as subscribers for new media requests.

2. Singleton: Changed Admin class to singleton as we wanted there to only be one admin instance.

We also made some changes by adding some variables to classes that we realized we needed when we began to implement the classes and database (ex. mediaID).