Book Shopping Application

Analysis and Design

Submitted By : Pallavi Madhuranath
Student ID : J5294158
Date : Dec 15 2015

Table of Contents

1. Introduction

This is an Book Shopping Application. It allows customer to select books from list of available books. Lets them add books to the cart. Further their purchase is processed and bill is generated. On the other hand administrator is allowed to add new book to the list and also delete book from the list. Current system is designed only for One customer and one administrator.

2. Requirements

1. The application should give an option of payment by card. Here card system is done only for demo purpose. Does not support any card authentication process.
2. Application should allow administrator (currently support one administrator) to add/delete books to/from database.
3. So the application should have logins for customer and administrator.
4. The application should hold a collection of entries regarding name of the book, author, price in a database .
5. The application should display to the customer (currently support one customer), book name, code and price.The user must be able to select items from the available list of books, add /delete to/from the cart.
6. Once shopping is done, customer should be able to proceed for billing.

3. Use Cases

3.1 Use Case Brief

Customer

Customer selects books to purchase and then adds them to the cart. Books added to the cart can also be removed from it. The customer will provide payment details. And then exits from the application.

Administrator

Administrator adds a new book to the database, delete a book from the database.

3.2 Fully dressed use case

Use cases for Online book shopping system

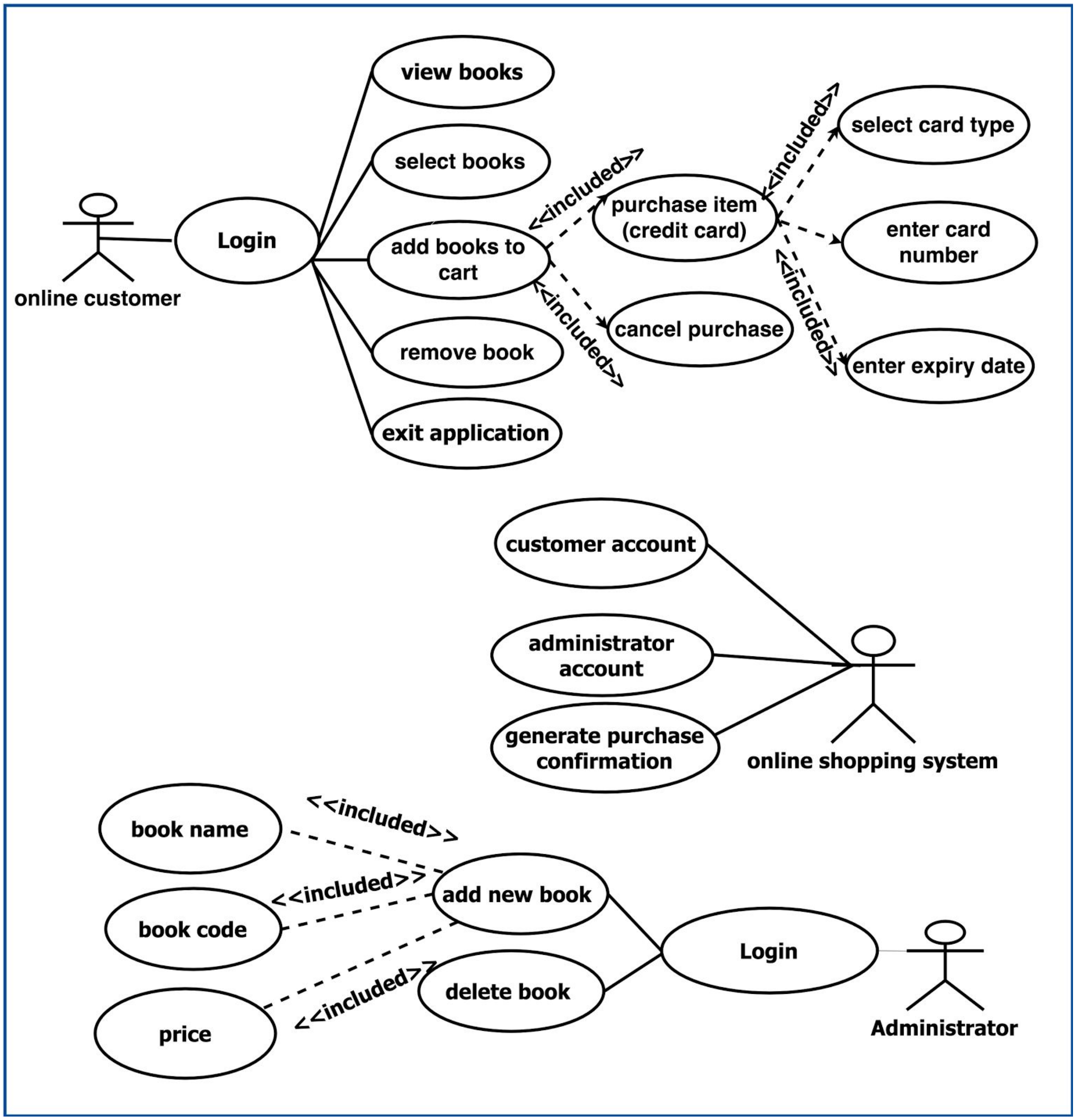


Figure 1 : Use case Diagram

3.3 Use case Table

Use Case No:	1
Scope:	Customer login
Actor	customer

Stakeholders and Interest:	**customer** : customer has to login to account in order to purchase books. **shopping system** : user account is created inside the system
Precondition:	A customer id and password is already saved in the system.
Postcondition:	search for books
Main success scenario:	1. Customer opens the website of book online shopping system. 2. Enters Login id and password.

Use Case No:	2
Scope:	view books
Actor	online customer
Stakeholders and Interest:	**customer** : search for books from the list **shopping system** : display list of books
Precondition:	customer should be logged in.
Postcondition:	select book/books
Main success scenario:	1.system displays list of book name, book code and size. 2. selects the required books 3. proceeds to add them to cart.
Alternate flow	Alternate scenario 1 : exit from application.

Use Case No:	3
Scope:	add to cart

Actor	online customer
Stakeholders and Interest:	**customer** : adds selected book/books to cart.
Precondition:	select book/books
Postcondition:	proceed for billing
Main success scenario:	1. customer adds selected book to the cart. [Alternate scenario 1] 2 proceeds for billing. [Alternate scenario 2]. 3. generates total bill.
Alternate flow	Alternate scenario 1 : remove book/books from the cart. Alternate scenario 2 : exit from application

Use Case No:	4
Scope:	purchase book
Actor	online customer
Stakeholders and Interest:	**customer** : purchase book **shopping system** : billing process, credit card process.
Precondition:	add books to cart. And card details already authenticated.
Postcondition:	system generates purchase confirmation
Main success scenario:	1. customer proceeds to purchase books 2. selects credit card type. 3. enters card number and expiry date.
Alternate flow	Alternate scenario 1 :cancels purchase.

Use Case No:	5
Scope:	administrator login
Actor	administrator
Stakeholders and Interest:	**administrator :** has to login to modify database.
Precondition:	admin id and password is authenticated and stored.
Postcondition:	modifies database.
Main success scenario:	1.admin logs in to his account.

Use Case No:	6
Scope:	modify database
Actor	administrator
Stakeholders and Interest:	**administrator** : wants to change information inside book database
Precondition:	admin login, database should be present.
Postcondition:	updated database
Main success scenario:	1. administrator chooses to add/delete information inside a database.
Alternate flow	If changing of database is failed. System informs administrator changing failed. Try again.

4. Sequence Diagrams

4.1 Customer

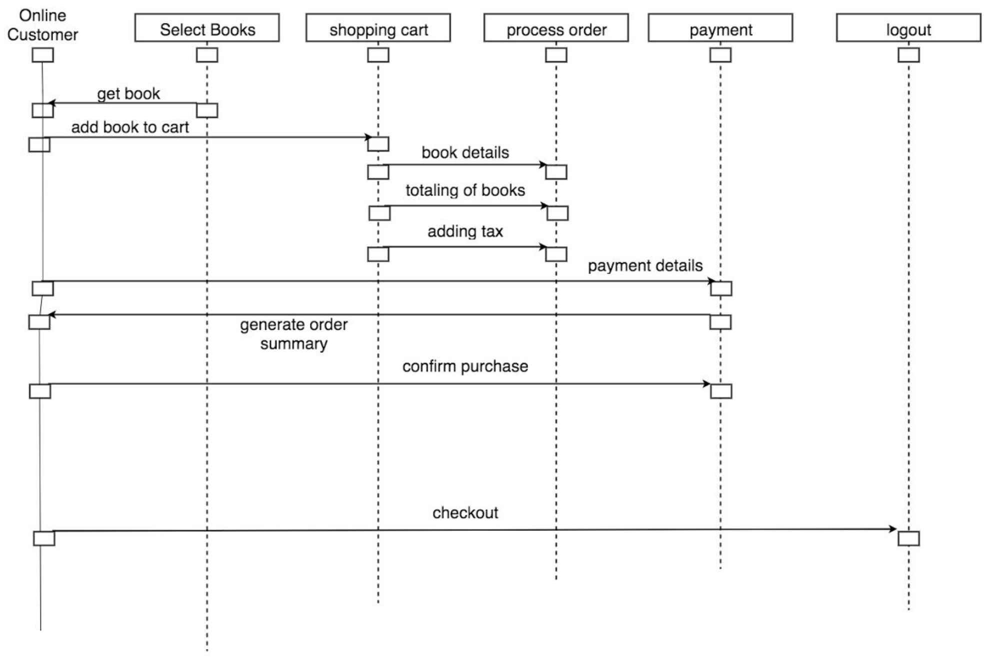


Figure 2 : Sequence diagram of customer use case

4.2 Administrator

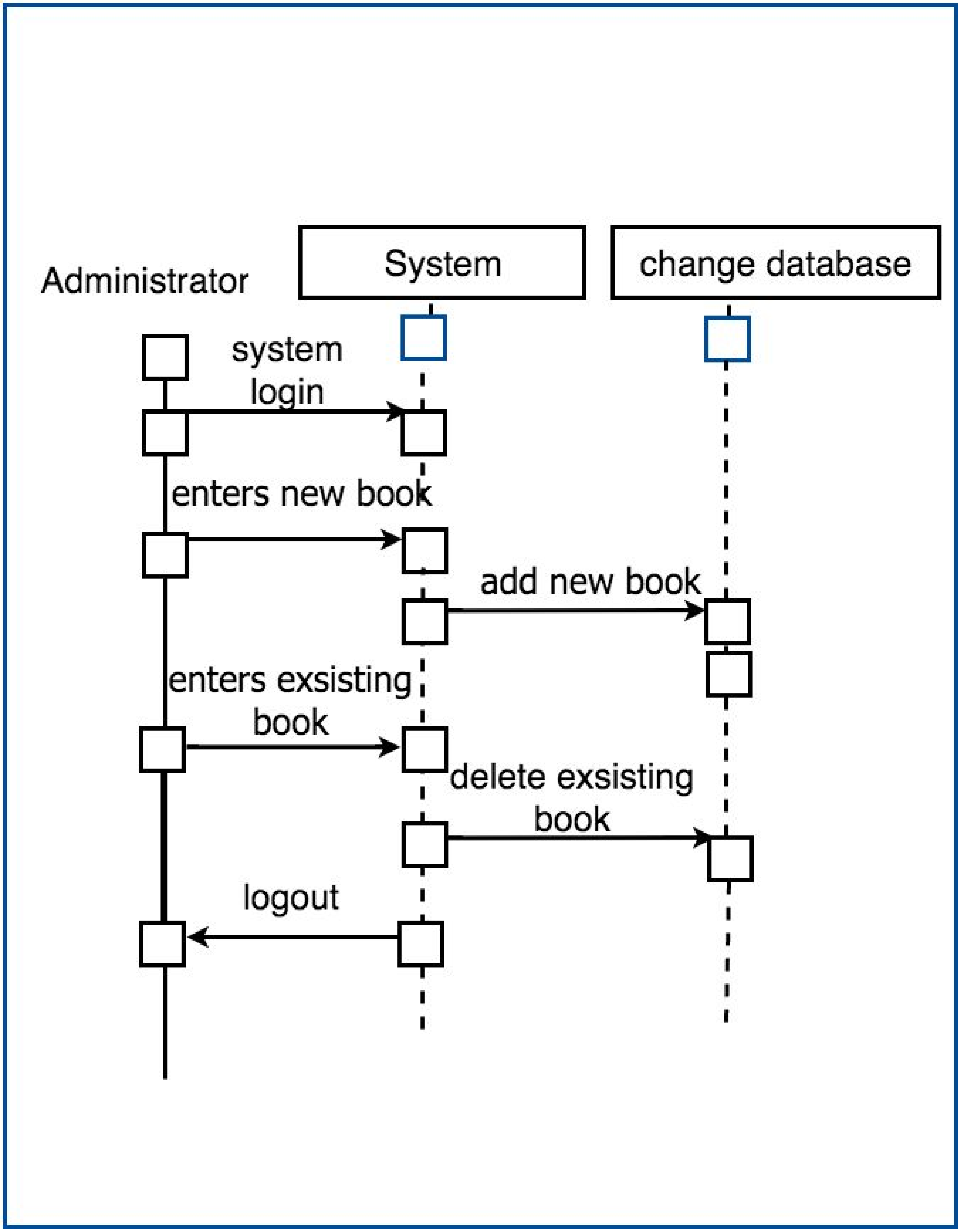


Figure 3 : Sequence diagram of administrator

5. Architecture Diagram

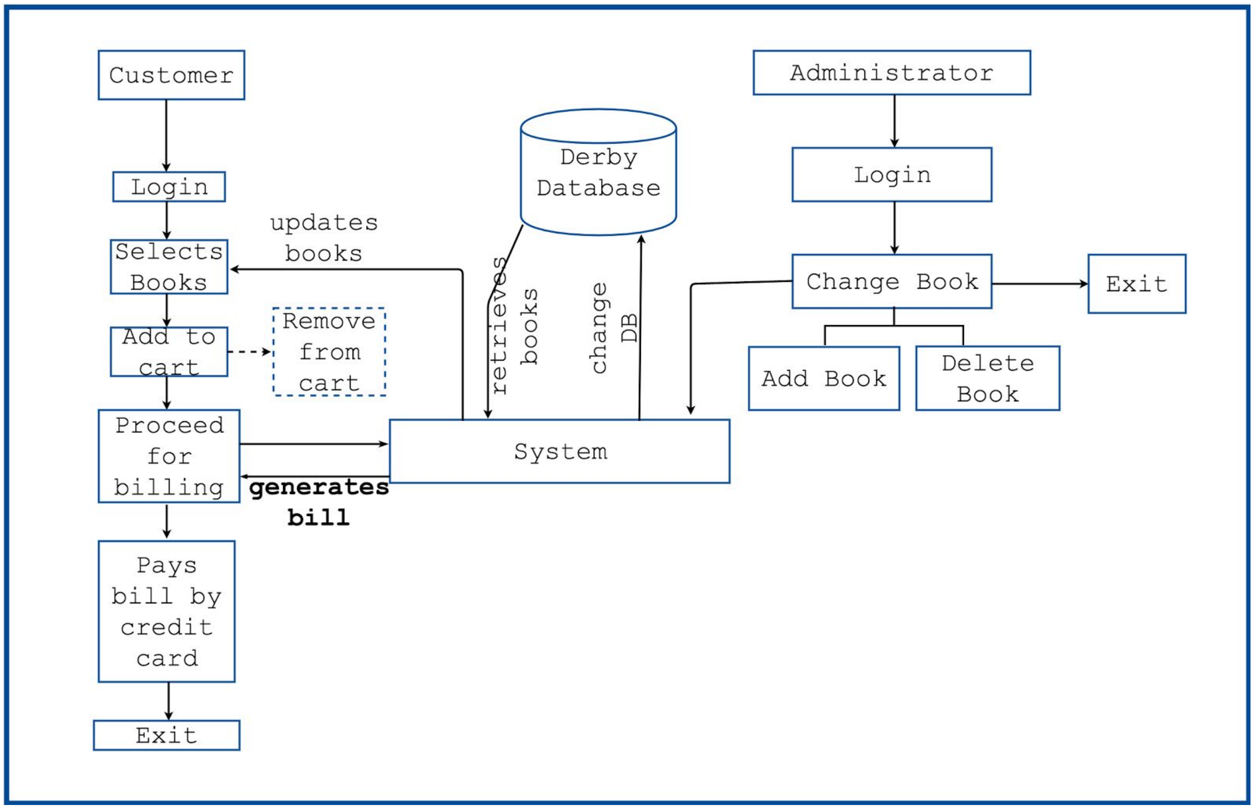


Figure 4 : **Architectural Diagram Top Level**

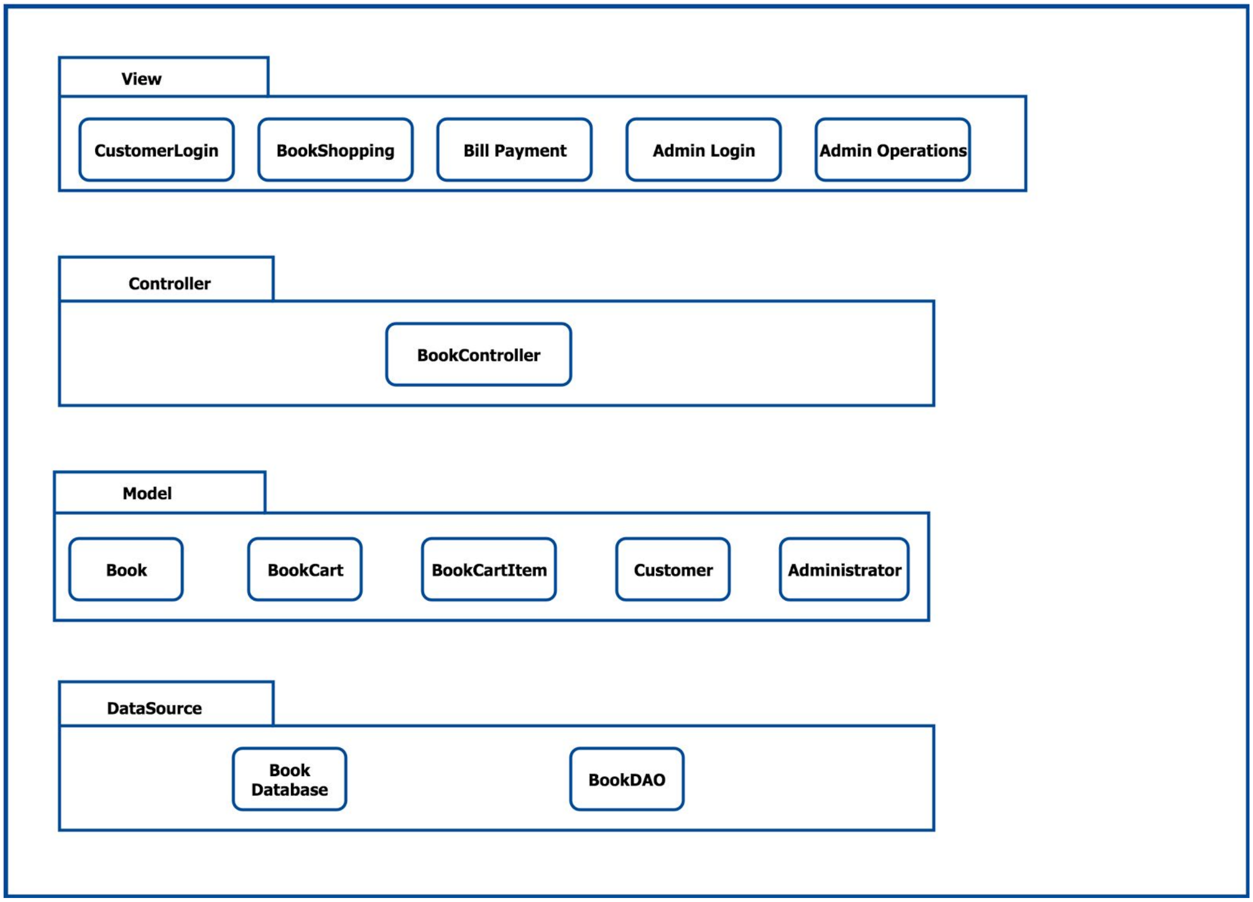


Figure 5 : Logical Architecture Diagram

5.1 Package "View"

Name	***Description***
CustomerLogin	GUI component that allows user to login to account.
BookShopping	GUI component that gives customer the list of books available to select from. Once book is selected it displays total price. Using this interface customer can add book to cart.
BillPayment	GUI components allows user to make payment by credit card.
AdminLogin	GUI component that allows admin to login to account.
AdminOperations	GUI component that allows admin to add/remove books from the system.

5.4 Package “controller”

Name	***Description***
BookController	This class represents a BookController which gets data from the dataSource and passes it to the view. It executes the actions on model as requested by the view.

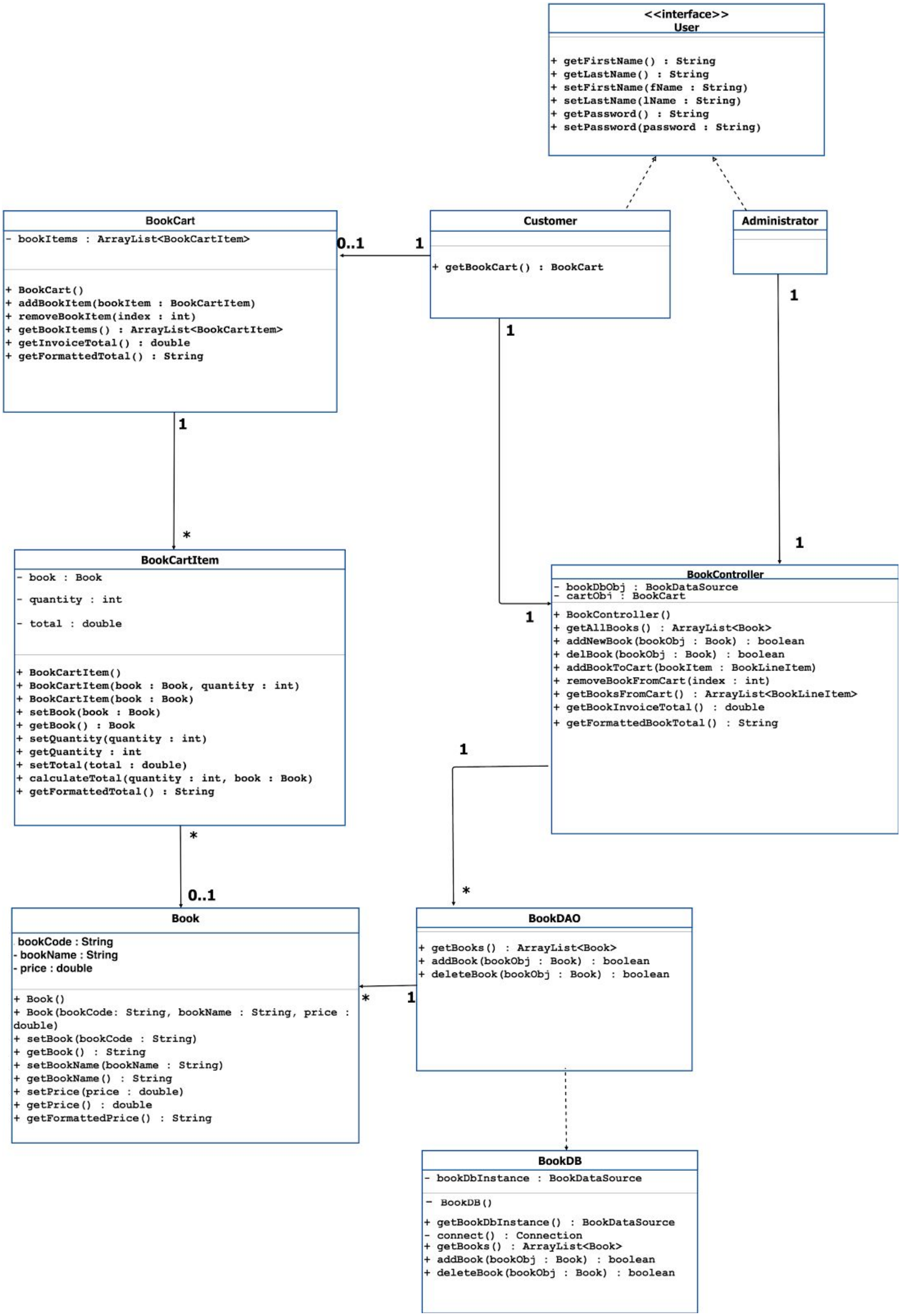
5.3 Package “model”

Name	***Description***
Book	This class holds all book attributes like book name, book code and price.
BookCart	This class represents a book cart to which books are added or removed.
BookCartItem	This class represents a book inside book cart.
User	This is an interface that represents an user and its properties.
Customer	This class represents a customer. Customer is a user who can add books to his book cart.
Administrator	This class represents a administrator. Administrator is a user who can add/remove books from the system.

5.4 Package “dataSource”

Name	***Description***
BookDB	This class implements interface BookDAO. It has query operations such as addBooks, deleteBooks and getBooks.
BookDAO	This is an interface that provides access to underlying derby database.

6. UML Class Diagram


<<interface>>
User
+ getFirstName() : String
+ getLastName() : String
+ setFirstName(fName : String)
+ setLastName(lName : String)
+ getPassword() : String
+ setPassword(password : String)
BookCart
- bookItems : ArrayList<BookCartItem>
+ BookCart()
+ addBookItem(bookItem : BookCartItem)
+ removeBookItem(index : int)
+ getBookItems() : ArrayList<BookCartItem>
+ getInvoiceTotal() : double
+ getFormattedTotal() : String
0..1
1
Customer
+ getBookCart() : BookCart
Administrator
1
1
1
*
BookCartItem
- book : Book
- quantity : int
- total : double
+ BookCartItem()
+ BookCartItem(book : Book, quantity : int)
+ BookCartItem(book : Book)
+ setBook(book : Book)
+ getBook() : Book
+ setQuantity(quantity : int)
+ getQuantity : int
+ setTotal(total : double)
+ calculateTotal(quantity : int, book : Book)
+ getFormattedTotal() : String
1
BookController
- bookDbObj : BookDataSource
- cartObj : BookCart
1
+ BookController()
+ getAllBooks() : ArrayList<Book>
+ addNewBook(bookObj : Book) : boolean
+ delBook(bookObj : Book) : boolean
+ addBookToCart(bookItem : BookLineItem)
+ removeBookFromCart(index : int)
+ getBooksFromCart() : ArrayList<BookLineItem>
+ getBookInvoiceTotal() : double
+ getFormattedBookTotal() : String
1
*
*
0..1
Book
bookCode : String
- bookName : String
- price : double
+ Book()
+ Book(bookCode: String, bookName : String, price : double)
+ setBook(bookCode : String)
+ getBook() : String
+ setBookName(bookName : String)
+ getBookName() : String
+ setPrice(price : double)
+ getPrice() : double
+ getFormattedPrice() : String
*
1
BookDAO
+ getBooks() : ArrayList<Book>
+ addBook(bookObj : Book) : boolean
+ deleteBook(bookObj : Book) : boolean
BookDB
- bookDbInstance : BookDataSource
- BookDB()
+ getBookDbInstance() : BookDataSource
- connect() : Connection
+ getBooks() : ArrayList<Book>
+ addBook(bookObj : Book) : boolean
+ deleteBook(bookObj : Book) : boolean