ReSource
Analysis Model

Submitted to:

Asst. Prof. Ma. Rowena C. Solamo
Faculty Member
Department of Computer Science
College of Engineering
University of the Philippines, Diliman

Submitted by:
Jaira Mathena B. Angeles
Danilo M. Mendoza II
Beatrice Pauline R. Mercado

In partial fulfillment of Academic Requirements
for the course
CS 191 Software Engineering I
of the
1st Semester, AY 2019-2020

Unique Reference:

The documents are stored in the https://github.com/mathena828/ReSource referenced with 2-Resource-Analysis Model.pdf.

Purpose:

This document will outline the necessary features that must be incorporated into the final web application, as well as guide readers on the various ways they can expect to interact with the system. It illustrates this information through a use case diagram. It also contains brief descriptions of the scope and limitations of each class therein.

Audience:

The target audience of this document is the CS 191 and CS 192 Software Engineering classes of the academic year 2019 to 2020, as well as other individuals who may need to evaluate the functional requirements of this project. The members of Group No. 2, namely, Jaira Mathena B. Angeles, Danilo M. Mendoza II, and Beatrice Pauline R. Mercado will be creating and referencing this model. Professor Ma. Rowena C. Solamo will be evaluating the contents of this submission to ensure that it fulfills the criteria for this academic requirement.

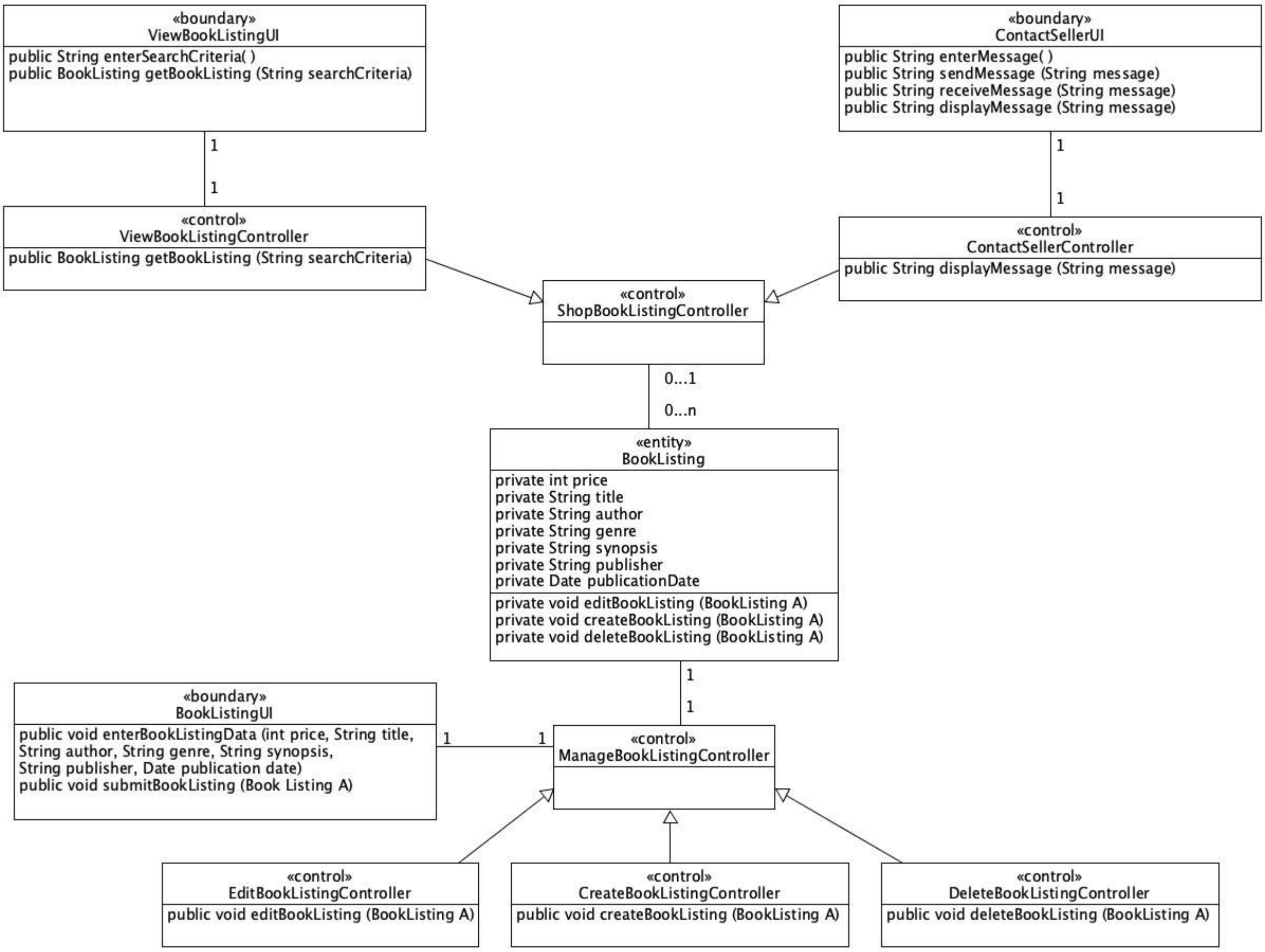
Revision Control:

Revision Date	*Person Responsible*	*Version Number*	*Modification*
10/01/19	Jaira Mathena Angeles	1.0	She created the analysis model. She defined the boundary, control, and entity classes.
10/03/19	Beatrice Pauline Mercado	1.1	She worked on the behavioral model for the Manage Book Listing use case.
10/03/19	Danilo Mendoza II	1.2	He worked on the behavioral model for the Shop Book Listing use case.

System Name: ReSource Book Trading System

Description: This system describes the way transactions are facilitated on the web application ReSource. It described the respective actions that buyers and sellers can perform on this platform. The buyers and sellers are both able to contact one another in order to coordinate the details of the exchange. Buyers are, to be precise, able to search for a specific book listings and set-up a chat room with the seller of said product. Likewise, sellers can manage book listings by maintaining their own inventories. This chain of events represent how product listings are posted by sellers and ultimately purchased by buyers within the system.

Analysis Model:


«boundary»
ViewBookListingUI
public String enterSearchCriteria( )
public BookListing getBookListing (String searchCriteria)
1
1
«control»
ViewBookListingController
public BookListing getBookListing (String searchCriteria)
«boundary»
ContactSellerUI
public String enterMessage( )
public String sendMessage (String message)
public String receiveMessage (String message)
public String displayMessage (String message)
1
1
«control»
ContactSellerController
public String displayMessage (String message)
«control»
ShopBookListingController
0...1
0...n
«entity»
BookListing
private int price
private String title
private String author
private String genre
private String synopsis
private String publisher
private Date publicationDate
private void editBookListing (BookListing A)
private void createBookListing (BookListing A)
private void deleteBookListing (BookListing A)
1
1
«boundary»
BookListingUI
public void enterBookListingData (int price, String title,
String author, String genre, String synopsis,
String publisher, Date publication date)
public void submitBookListing (Book Listing A)
1
1
«control»
ManageBookListingController
«control»
EditBookListingController
public void editBookListing (BookListing A)
«control»
CreateBookListingController
public void createBookListing (BookListing A)
«control»
DeleteBookListingController
public void deleteBookListing (BookListing A)

Boundary Classes:

Class Name	**Description**
BookListingUI	This is the interface of the seller to the system whenever they need to manage a book listing within their personal inventory. *Responsibilities*: public void enterBookListing (int price, String title, String author, String genre, String synopsis, String publisher, Date publicationDate) public void submitBookListing (BookListing A)
ViewBookListingUI	This is the interface of the buyer to the system whenever they need to view a book listing from the ReSource database. *Responsibilities*: public String enterSearchCriteria ( ) public BookLisitng getBookListing (String searchCriteria)
ContactSellerUI	This is the interface of the buyer to the system whenever they need to contact the seller to inquire about a specific book listing. *Responsibilities*: public String enterMessage ( ) public String sendMessage (String message) public String receiveMessage (String message) public String displayMessage (String message)

Control Classes:

Class Name	**Description**
ManageBookListingController	This is the control that manages book listings. It is considered an abstract class.
EditBookListingController	This is the control that adds a book listing to the system. It extends ManageBookListingContoller. *Responsibilities*: public void editBookListing (BookListing A)
CreateBookListingController	This is the control that updates a book listing in the system. It extends ManageBookListingContoller. *Responsibilities*: public void createBookListing (BookListing A)
DeleteBookLisitngController	This is the control that removes a book listing from the system. It extends ManageBookListingContoller. *Responsibilities*: public void deleteBookListing (BookListing A)
ShopBookListingController	This is the control that shops book listings. It is considered an abstract class.
ViewBookListingController	This is the control that searches for book listings in the system. It extends ShopBookListingContoller. *Responsibilities*: public BookListing getBookListing (String searchCriteria)
ContactSellerController	This is the control that facilitates the exchange of books through the system. It extends ShopBookListingContoller. *Responsibilities*: public String displayMessage (String message)

Entity Classes:

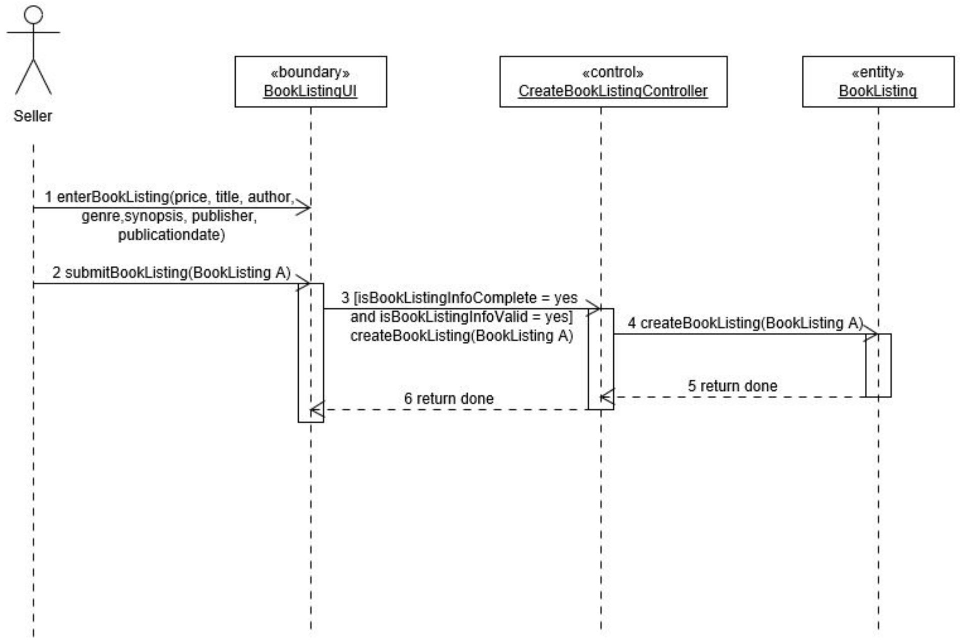
Class Name	**Description**
BookListing	This is the entity class book listing, which contains the data about the book listing. *Attributes:* private int price private String title private String author private String genre private String synopsis private String publisher private Date publicationDate

Behavioral Model:

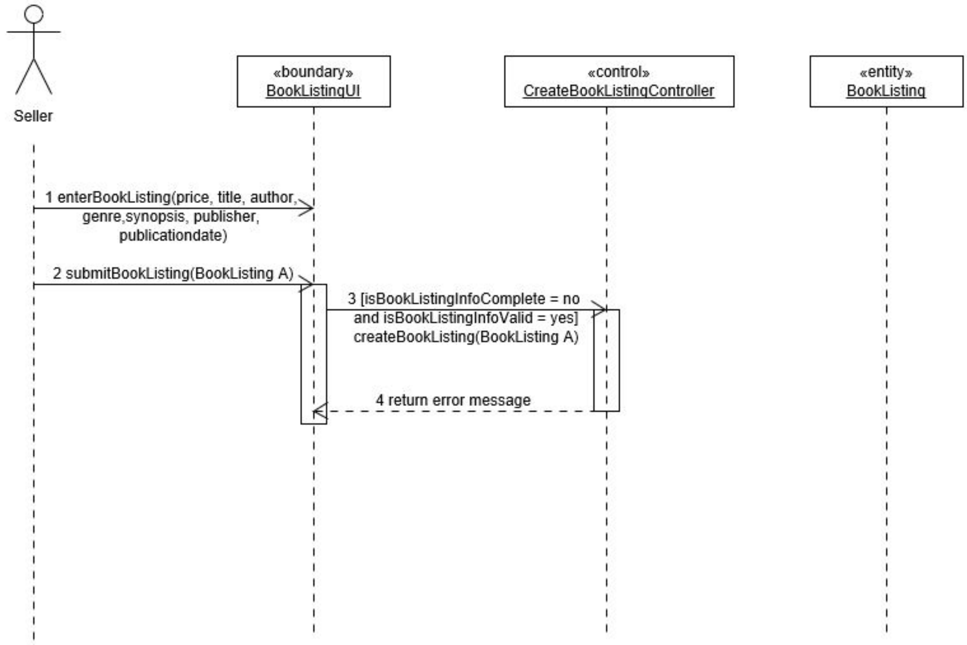
***Use-Case Name*:** 1.0 Manage Book Listing

Description: This pertains to the seller's ability to maintain their own inventory of book listings. They can do so through three features, Edit Book Listing (Use-Case 1.1), Create Book Listing (Use-Case 1.2), and Delete Book Listing (Use-Case 1.3). The seller's posts are collated on the site so that they can be found by potential buyers.

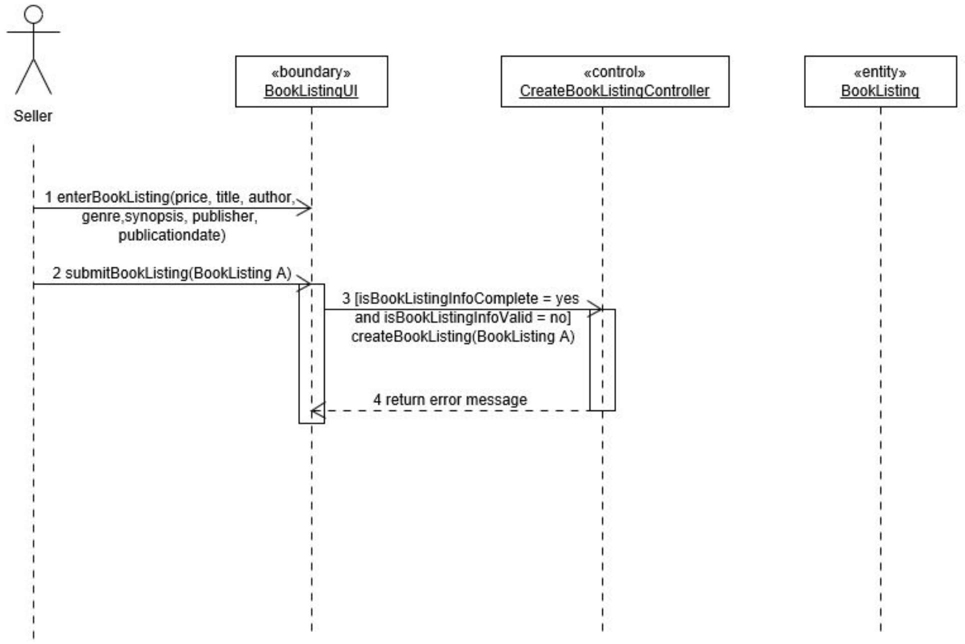
Scenario 1: A book listing is successfully posted by the seller. (Basic Flow)


Seller
«boundary»
BookListingUI
«control»
CreateBookListingController
«entity»
BookListing
1 enterBookListing(price, title, author, genre,synopsis, publisher, publicationdate)
2 submitBookListing(BookListing A)
3 [isBookListingInfoComplete = yes and isBookListingInfoValid = yes] createBookListing(BookListing A)
4 createBookListing(BookListing A)
5 return done
6 return done

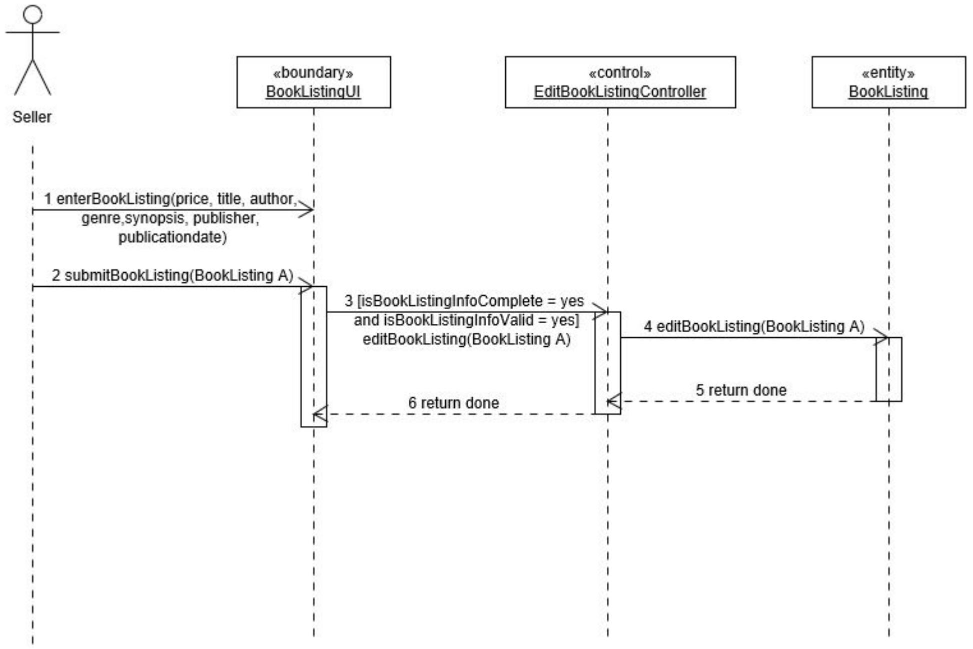
Scenario 2: A book listing cannot be posted by the seller due to incomplete data.


Seller
«boundary»
BookListingUI
«control»
CreateBookListingController
«entity»
BookListing
1 enterBookListing(price, title, author, genre,synopsis, publisher, publicationdate)
2 submitBookListing(BookListing A)
3 [isBookListingInfoComplete = no and isBookListingInfoValid = yes] createBookListing(BookListing A)
4 return error message

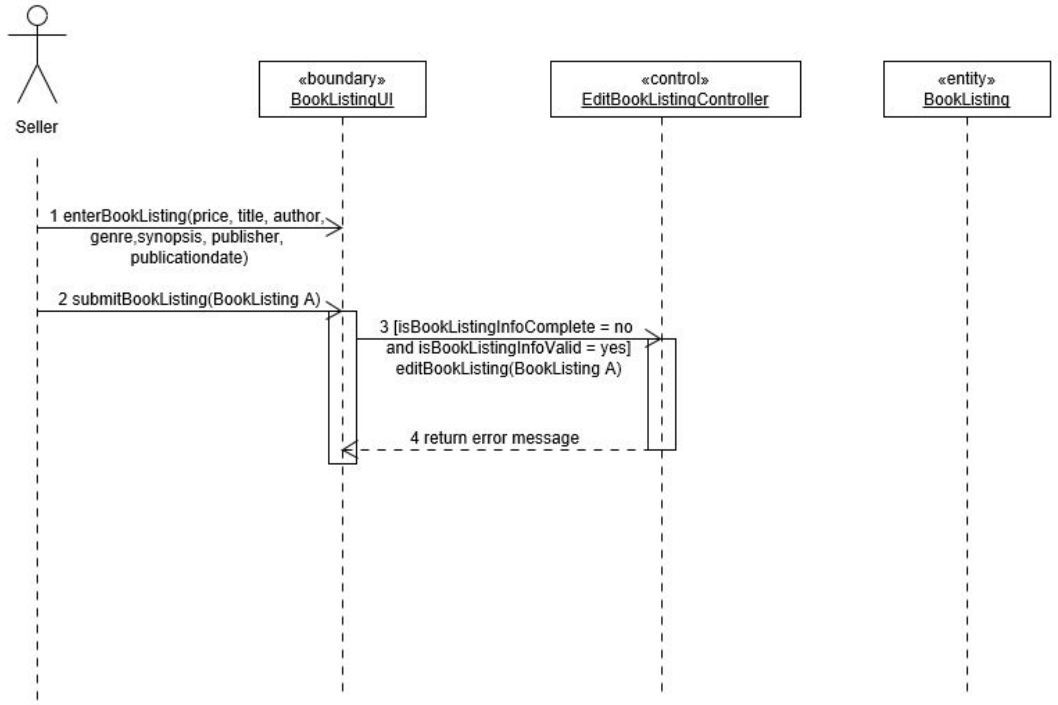
Scenario 3: A book listing cannot be posted by the seller due to invalid data.


Seller
«boundary»
BookListingUI
«control»
CreateBookListingController
«entity»
BookListing
1 enterBookListing(price, title, author, genre,synopsis, publisher, publicationdate)
2 submitBookListing(BookListing A)
3 [isBookListingInfoComplete = yes and isBookListingInfoValid = no] createBookListing(BookListing A)
4 return error message

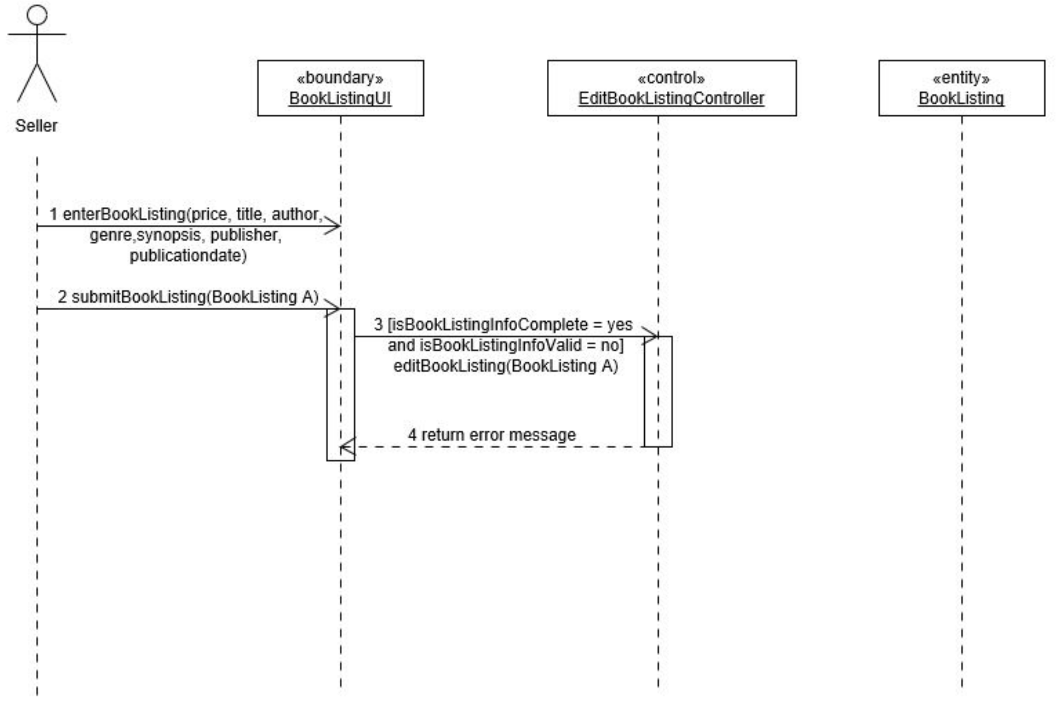
Scenario 4: The book listing information successfully is updated. (Basic Flow)


Seller
«boundary»
BookListingUI
«control»
EditBookListingController
«entity»
BookListing
1 enterBookListing(price, title, author, genre,synopsis, publisher, publicationdate)
2 submitBookListing(BookListing A)
3 [isBookListingInfoComplete = yes and isBookListingInfoValid = yes] editBookListing(BookListing A)
4 editBookListing(BookListing A)
5 return done
6 return done

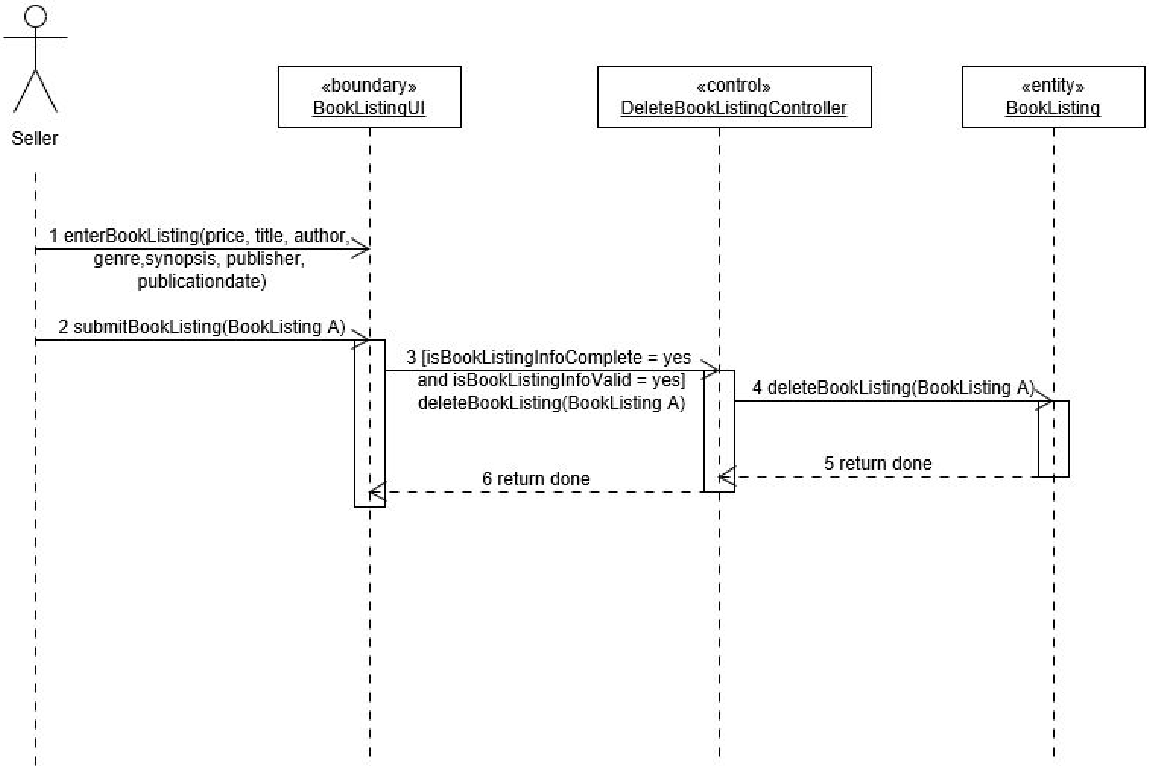
Scenario 5: The book listing information cannot be updated due to incomplete data.


Seller
«boundary»
BookListingUI
«control»
EditBookListingController
«entity»
BookListing
1 enterBookListing(price, title, author, genre,synopsis, publisher, publicationdate)
2 submitBookListing(BookListing A)
3 [isBookListingInfoComplete = no and isBookListingInfoValid = yes] editBookListing(BookListing A)
4 return error message

Scenario 6: The book listing information cannot be updated due to invalid data.


Seller
«boundary»
BookListingUI
«control»
EditBookListingController
«entity»
BookListing
1 enterBookListing(price, title, author, genre,synopsis, publisher, publicationdate)
2 submitBookListing(BookListing A)
3 [isBookListingInfoComplete = yes and isBookListingInfoValid = no] editBookListing(BookListing A)
4 return error message

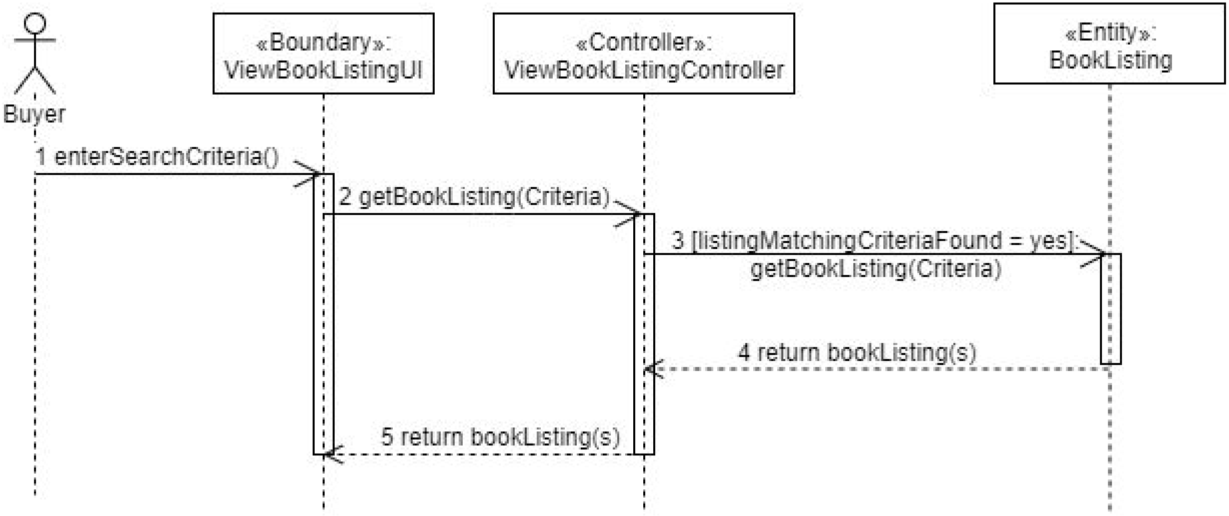
Scenario 7: A book has been sold to a buyer.


Seller
«boundary»
BookListingUI
«control»
DeleteBookListingController
«entity»
BookListing
1 enterBookListing(price, title, author, genre,synopsis, publisher, publicationdate)
2 submitBookListing(BookListing A)
3 [isBookListingInfoComplete = yes and isBookListingInfoValid = yes] deleteBookListing(BookListing A)
4 deleteBookListing(BookListing A)
5 return done
6 return done

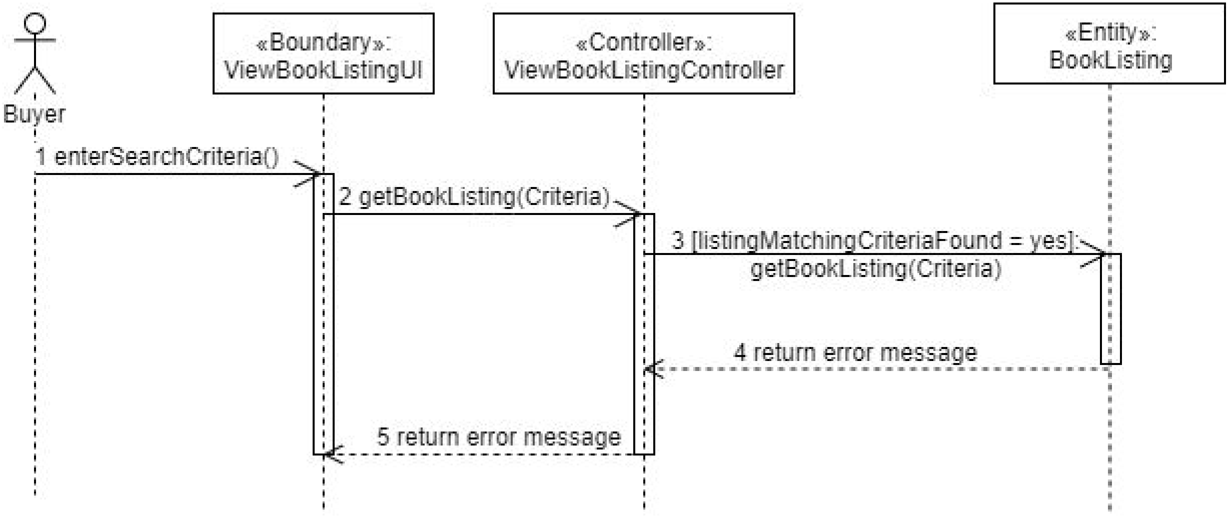
***Use-Case Name*:** 2.0 Shop Book Listing

Description: This pertains to the buyer's ability to respond to a book listings posted by the buyers on the platform. This encompasses two key operations. Namely, these are View Book Listing (Use Case 2.1) and Contact Seller (Use Case 2.2).

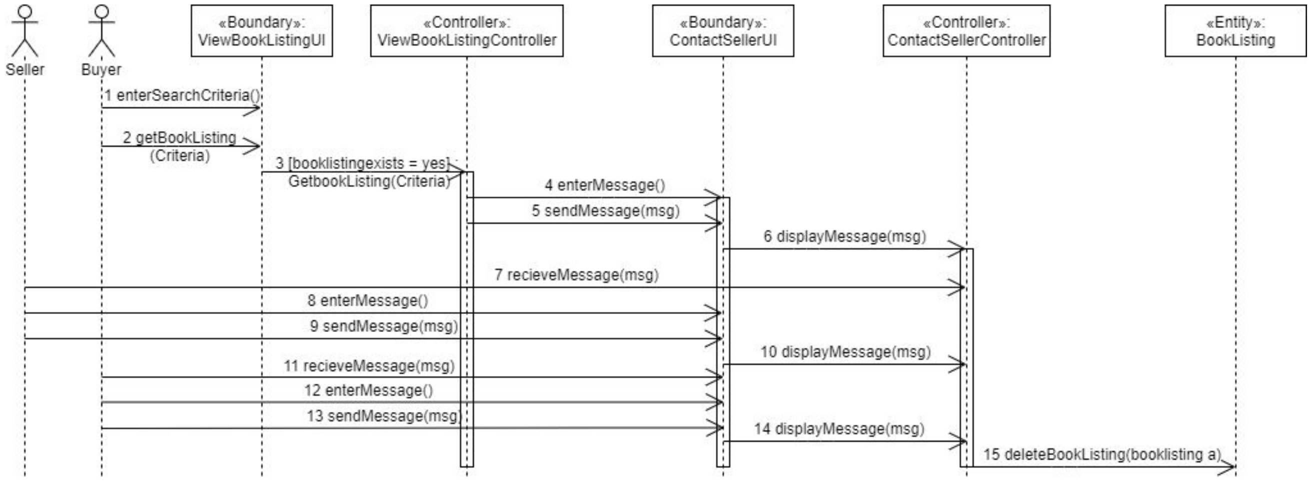
Scenario 1: The buyer finds the book they are searching for. (Basic Flow)


Buyer
«Boundary»: ViewBookListingUI
«Controller»: ViewBookListingController
«Entity»: BookListing
1 enterSearchCriteria()
2 getBookListing(Criteria)
3 [listingMatchingCriteriaFound = yes]: getBookListing(Criteria)
4 return bookListing(s)
5 return bookListing(s)

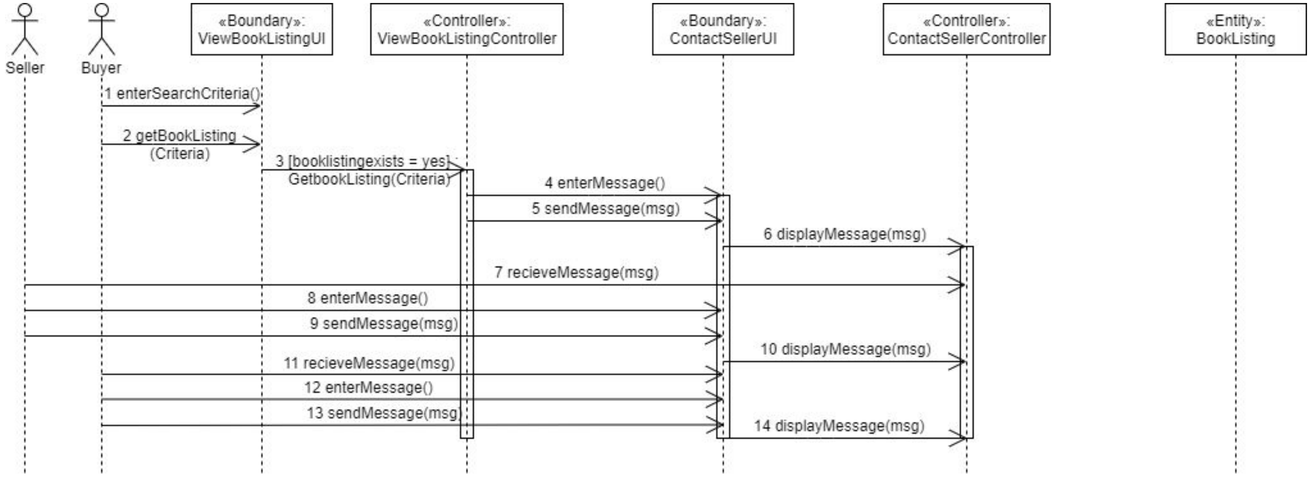
Scenario 2: The book listing the buyer is searching for does not exist.


Buyer
«Boundary»: ViewBookListingUI
«Controller»: ViewBookListingController
«Entity»: BookListing
1 enterSearchCriteria()
2 getBookListing(Criteria)
3 [listingMatchingCriteriaFound = yes]: getBookListing(Criteria)
4 return error message
5 return error message

Scenario 3: The buyer places then purchases an order (Basic Flow)



Scenario 4: The buyer places then cancels an order.


Seller
Buyer
«Boundary»: ViewBookListingUI
«Controller»: ViewBookListingController
«Boundary»: ContactSellerUI
«Controller»: ContactSellerController
«Entity»: BookListing
1 enterSearchCriteria()
2 getBookListing (Criteria)
3 [booklistingexists = yes] GetbookListing(Criteria)
4 enterMessage()
5 sendMessage(msg)
6 displayMessage(msg)
7 recieveMessage(msg)
8 enterMessage()
9 sendMessage(msg)
10 displayMessage(msg)
11 recieveMessage(msg)
12 enterMessage()
13 sendMessage(msg)
14 displayMessage(msg)