Software Requirements and Design Specification For BOOKS R US

Faculty: Mr. Kiran Chittargi
Term: Fall 2018

Team Members:

Ming Lei
Stephen Smeal
Graciela Casebeer

Table of Contents

Revision History

Version	Date	Name	Description
0.1	10/01/2018	Graciela Casebeer	Project Proposal – first draft
0.2	10/05/2018	Ming Lei Stephen Smeal	Modified use case Added classes and JSP outline. Added Screenshot.
0.3	10/05/2018	Graciela Casebeer	Created Appendix A to hold all screenshots
0.4	10/06/2018	Ming Lei Stephen Smeal Graciela Casebeer	Updated JavaBean classes. Added database diagram section. Added database and helper classes section.
0.5	10/06/2018	Graciela Casebeer	JavaBean classes diagram added. Database diagram added.
0.6	10/07/2018	Ming Lei	Added additional screenshot: Registration Validation, Registration Success, Registration Failed, User Login, Product Information, Cart
1.0	10/09/2018	Ming Lei Stephen Smeal Graciela Casebeer	Preliminary proposal. Submittal ready version.

Introduction

Overview

The Books R Us application will be a web application accessible through a web browser. The application will provide access to the Books R Us e-store where shoppers will be able to browse the store catalog and purchase the titles of their preference. The application will also provide access to Books R Us employees and give them the ability to edit the store catalog (e.g. add/modify/delete catalog items).

This document provides information on the requirements for the Books R Us software application. Project goals, scope, and definitions are given in the introduction. Design constraints and application environment are described in the following section. Functional and non-functional requirements are outlined to show the system features and expected user interaction. The document ends with the system architecture section, containing the implementation technical details.

Goals and Objectives

The main goal of this application is to give Books R Us the ability to do commercial transactions electronically on the internet. The application will provide the following:

1. An electronic storefront to sell direct to consumers (i.e. web site).
2. A portal for Books R Us employees to edit the store catalog.
3. The web site should:
    a. Have a clutter-free look and feel.

b. Function in a user-friendly and intuitive manner.

Scope

The Books R Us application will provide a mechanism for customers to purchase books offered in the e-store. A user can freely browse the product offering and read descriptions, prices, and customer reviews.

The application will support two user roles: customer (basic feature) and administrator (stretch feature).

Customers are individuals interested in purchasing the products offered in the e-store. Customers interested in placing an order must have a profile. A profile is created by registering at the web site. The registration process creates a username and password to identify each customer. To place an order, the user must login to the site with their username and password. Upon successful login, the user can add items to his or her cart, determine the quantity of items desired to purchase, and add reviews for a product.

Administrators are employees working for Books R Us who are responsible for editing product profiles and customer reviews.

Multiple users – customers and administrators – should be able to login simultaneously.

Project Team and Work Breakdown

The project team to implement this application consists of three engineers. Their names and high-level areas of responsibilities are listed below. This is a flexible division of work in which any team member could help in any area of development as necessary to meet our deadlines or to fulfill an area of interest to the engineer.

- Ming Lei: primary focus on view module
  - Graphics
  - JSPs
  - Sample data
- Stephen Smeal: primary focus on model, database design, and database interaction.
  - JavaBeans
  - Database access layer classes
  - Database connection
- Graciela Casebeer: primary focus on controller module.
  - Controller servlet and sub-servlets

Definitions

Application – what is being described here; the software system specified in this document.

Project – activities that will lead to the production of the application described here.

User – the person or persons who will actually interact with the application.

Customer – the person who intends to purchase a book through the application.

Consumer – the person who intends to purchase a book through the application.

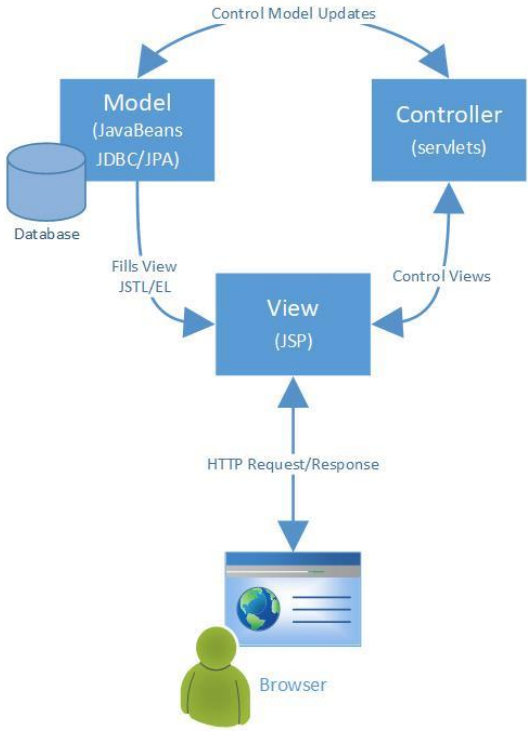
Administrator – a Books R Us employee who can add/delete/modify product profiles and customer reviews in the application.
Use case – describes a goal-oriented interaction between the system and the user.
Role – category of users that share similar characteristics.
Controls – the individual elements of a user interface. For example: button, checkbox, dropdown list.
Product(s) – the items for sale through the application.
Basic feature – the feature will be part of the release.
Stretch feature – the feature will be part of the release if time allows.

General Design Constraints

Books R US Application Environment

The application will follow the Model-View-Controller pattern separating the code in three layers: the model, the view, and the controller. The model, defining the business layer of the application, will use JavaBeans. The view, defining the user interface to the application via a browser, will use JSPs. The controller, in charge of managing the flow of the application, will use servlets. The system will have a database (MySQL). To access the database, the system will have model classes specifically designed for interaction with the database; these will be separate from the JavaBean model classes.

To the extent possible, the design will aim to keep the model, view, and controller modules as independent as possible. Below is a high level diagram describing the Model 2 (MVC) pattern.



User Characteristics

The application will support two user roles: customer (basic feature) and administrator (stretch feature).

- Customer is the individual consumer interested in purchasing the products offered in the e-store. Given that e-commerce is nowadays a very common way to trade products and services, it is expected that customers using the Books R Us application have previous e-store purchasing experiences.
  This is a basic feature and as such it must be part of the release.
- Administrators are employees working for Books R Us who are responsible for editing product profiles and customer reviews. The administrators are expected to be familiar with the store product offering, pricing, and inventory.
  This is a stretch feature and as such it will be part of the release if time allows.

Mandated Constraints

The application will use a browser to give access to end users. The primary browser supported should be Chrome. Support for other browsers such as Edge and Firefox is a secondary priority.

Nonfunctional Requirements

Operational Requirements

The user interface must be very intuitive. There will be no end user manual; therefore, usability is important for users to easily navigate and use the application.

Performance Requirements

The application should support multiple sessions from different users simultaneously. Because this is not a production application, we will aim to do our testing with at least 10 simultaneous sessions with a mix of user roles, for example: 8 customer and 2 administrator users.

Security Requirements

Any user can freely browse the web site in a read-only capacity to see the product offering, prices, and product reviews. However, if a customer wants to place an order or an administrator needs to edit a product, the user must login with their username and password. Upon successful login, all traffic should be encrypted and go through HTTPS. This is important to protect hacking of user profiles and purchasing data.

Documentation and Training

The application will be available by visiting to the store's website home page via a browser. There will be no documentation or training needed for end users; the application should be intuitive

External Interface

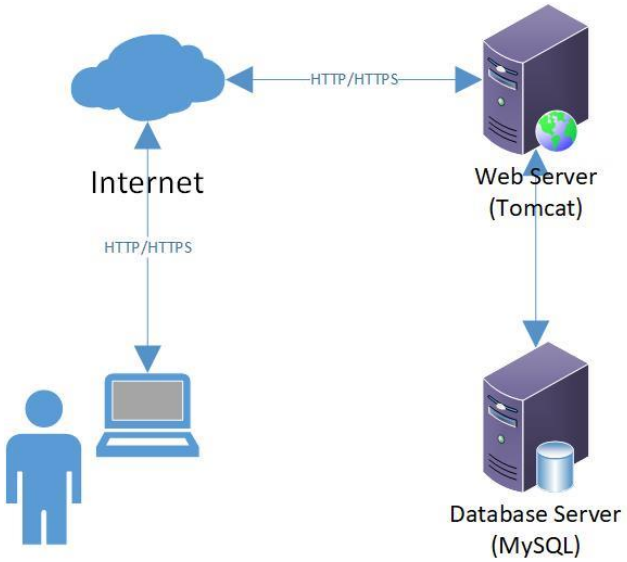
User Interface

The user interface will be minimalistic and visually appealing. Busy screens are a bit distracting and we want end users to be able to focus on the product offering. A clean and decluttered approach will guide the design.

The user interface should also be intuitive. The application should be easy to use without any training to the end user. Put in a more specific way, we want 80% of the users to be able to use 80% of the features within a few seconds without requiring any training.

Software Interface

The application will have remote software interface using web technologies such as HTTP/HTTPS.



Functional Requirements

The following use cases describe the functional requirements. Many of these cases describe the graphical user interface (GUI). To see the list of preliminary GUI screenshots for the application, please refer to Appendix A.

Use Case 1: Create an account

This use case begins when the user clicks on the “Register” link on top of the home page.This use case describes what happens when the user selects the create an account option.

Basic path

1. User clicks on “Register” option.
2. The application displays a screen asking for first name, last name, email, and password. This screen has two buttons: create and cancel.

3. The user enters his/her first name, last name, email, and password.
4. The user presses the create button.
5. The application verifies if the user email already exists in the database.
6. The email does not exist in the database.
7. A user account is created and a record added to the database.
8. The user is automatically logged in upon the account creation and a secure connection is established (e.g. https).
9. Screens available only to logged in users should now be accessible (e.g. checkout screen to review order and pay).
10. The application sends the original screen from where the user selected the *create an account* option.
11. The browser displays the page sent by the application, now using https.

Alternate path "a"

1. Same as basic path.
2. Same as basic path
3. Same as basic path.
4. Same as basic path.
5. Same as basic path.
6. The username or email exists in the database.
7. The application shows an error message page such as "A user already exists with the login <username> or < user email >".

Alternate path "b"

1. Same as basic path.
2. Same as basic path.
3. Same as basic path.
4. The user presses the cancel button.
5. The application sends the original screen from where the user selected create an account.

Use Case 2: User login

This use case begins when the user clicks on the "Sign in" link on top of homepage. This use case describes what happens when the user selects the *sign in* option.

Basic path

1. User clicks on *sign in* option.
2. The application prompts for username and password.
   Note: email is used as the username.
3. The user enters his/her email as the username.
4. The user enters his/her password.
5. The user clicks the submit button.
6. The browser sends the username and password information to the application.
7. The application verifies the username and password match with the database.
8. The application verifies the user role and sets it in the appropriate user javabean.
9. If the match is successful, a secure connection is established (e.g. https).
10. The application sends the original screen from where the user selected *sign in*.

11. The browser displays the page sent by the application, now using https.

Alternate path “a”

1. Same as basic path
2. Same as basic path
3. Same as basic path
4. Same as basic path
5. Same as basic path
6. Same as basic path
7. Same as basic path
8. If the match is not successful, the application shows a page with an error message such as “Incorrect username or password” and a button to “try again”.
9. The user pushes the “try again” button.
10. The login page is displayed.

Alternate path “b”

1. Same as basic path
2. Same as basic path
3. Same as basic path
4. Same as basic path
5. The user clicks the cancel button.
6. The application sends the original screen from where the user selected *sign in*.

Use Case 3: Browse a product category

This use case begins when the user clicks on one of the links for product categories. The click should display a page with a list of products offered for the selected category.

Basic path

1. User clicks on a *category* link.
2. Application displays a page containing the category product offering list.
3. Each product item in the page will have a product summary as follows:

< Book Title >
by < Author >
< Rating Stars> *(stretch feature if time allows)*
Price: < $xx.yy >

Use Case 4: View product details and add product to cart

This use case begins when the user clicks on a specific product from a given product category offering list.

Basic path

1. User clicks on a specific item in a given product category offering list.
2. Application displays a product page containing:

    a. Larger book cover view
    b. Book Title
    c. Book Author
    d. Rating Stars *(stretch feature if time allows)*
    e. Price
    f. Quantity (selectable drop-down field)
    g. Customer reviews text *(stretch feature if time allows)*
    h. Button to *add item* to the shopping cart
    i. Button to *view shopping cart*
    j. Button to *continue shopping*
3. User presses the *add item* button.
4. The product is added to the user's cart.

Alternative path "a"

1. Same as basic path.
2. Same as basic path.
3. User presses the *view shopping cart*.
4. The application displays the cart page.

Alternative path "b"

1. Same as basic path.
2. Same as basic path.
3. User presses the *continue shopping* button.
4. The application returns the user to the product category page that the specific product selected belongs to.

Use Case 5: View and edit shopping cart

This use case begins when the user clicks on the *view shopping cart* button in a given page, or when the user clicks the cart link at the top right corner of the application screen.

Basic path

1. User clicks on the *view shopping cart* button or the *cart* link at the top right corner of the application screen.
2. The application displays the cart page with the order summary and the *checkout* button. Every item in the order should have a *remove* button for the user to eliminate an item from the order. The *quantity* field for each item should be editable (e.g. drop down) so the user can adjust the order quantity. The order summary will contain one line per item added to the cart with the following information:
    a. Item description
    b. Item price
    c. Item quantity (*editable field*)
    d. Item subtotal (price x quantity)
    e. Grand total (all subtotal numbers added)
    f. *Remove* button

Alternative path “a”

1. Same as basic path.
2. Same as basic path.
3. The end user adjust the *quantity* field in any of the items in the cart.
4. The application re-calculates and displays the new subtotal and grand total numbers.

Alternative path “b”

1. Same as basic path.
2. Same as basic path.
3. The end user presses the *remove* button in any or the order items.
4. The application removes the item from the cart.
5. The application re-calculates and displays the new subtotal and grand total numbers.

Use Case 6: Checkout and submit (or cancel) an order

This use case begins when the user clicks on the cart page *checkout* button.

Basic path

1. User clicks on the cart page *checkout* button.
2. The application:
    a. Displays the order summary (described in user case 5 above)
    b. Prompts user to fill out shipping/billing address:
    Assumption: the shipping and billing address are the same.
    Assumption: the package ships attention to the user’s name and lastname.
        i. Street address
        ii. City
        iii. State
        iv. Zip code
    c. Prompts user to fill out billing information:
    Assumption: the name on card is the same as the user’s name and lastname.
        i. Card type
        ii. Card number
        iii. Expiration month
        iv. Expiration year
    d. Displays two buttons: *submit order* or *cancel*
3. User clicks on the *submit order* button to place the order.
4. The application sends an email to the user confirming the order and including the invoice.

Alternate path “a”

1. Same as basic path.
2. Same as basic path.
3. User clicks on the *cancel* button to cancel the order and return to the last shown page prior to the cart page.

Use Case 7: Edit product profile (stretch feature)

This is a stretch feature. It will be implemented if time allows.

This use case begins when an administrator user clicks on a particular product item and the system displays the product page. *Assumption:* the user successfully logged in and the system identified his/her role as an administrator.

Basic path

1. Administrator user clicks on a specific item in a given product category offering list.
2. Application displays a product page containing:
    a. Larger book cover view
    b. Book Title
    c. Book Author
    d. Rating Stars *(stretch feature if time allows)*
    e. Price
    f. Quantity (selectable drop-down field)
    g. Customer reviews text *(stretch feature if time allows)*
    h. Button to *edit* item
    i. Button to *save* item
    j. Button to cancel
3. Administrator user presses the *edit* item button.
4. The system enables read/write permissions on the fields described in # 2 above.
5. The administrator modifies any/all the fields.
6. The administrator clicks the *save* button.
7. The system updates the product profile with the new information.

Alternative path "a"

1. Same as basic path.
2. Same as basic path.
3. Same as basic path.
4. Same as basic path.
5. Same as basic path.
6. The administrator clicks the *cancel* button.
7. The system restores the original information prior to any modifications and makes the product profile fields read-only.

System Architecture

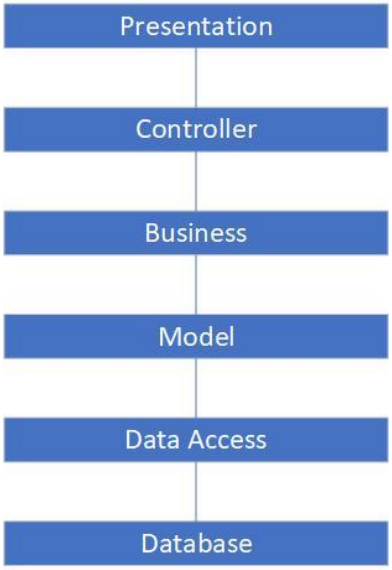
Introduction

The purpose of this document is to describe the architecture and design of the Books R Us e-commerce application.

Application Design

High-Level Design (Architecture)

The high-level view or architecture consists of six major components as shown in the diagram below. This architecture hierarchy allows modularity and flexibility.



The top layer is the *presentation layer*. It consists of the graphical user interface (GUI) where end users interact with the system. This layer will be implemented by a series of JSPs that are described later in this document.

The *controller layer* processes and responds to events triggered by user actions. It is also in charge of making changes on the model.

The *business layer* executes the business logic algorithms and calculations. Example: subtotal, taxes, and grand total calculations.

The *model layer* contains the classes that strictly consist of data (e.g. JavaBeans). Example: User data, Cart data, etc.

The *data access layer* is responsible to handle all database transactions and connectivity.

The *database layer* is the lowest level in the hierarchy and it is responsible for storing data in persistent storage. We believe we'll be using the MySQL database.

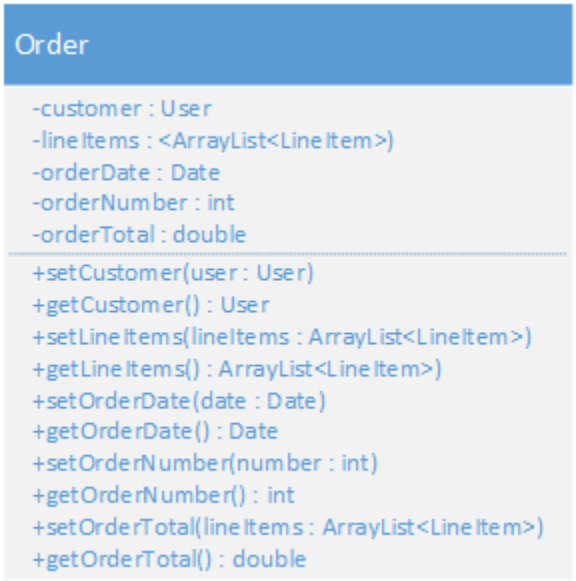
Mid-Level Design

This section explains the static and dynamic aspects of the system. We use class and sequence diagrams where appropriate to convey the design approach.

JavaBean Classes Outline

The class diagrams below show the five business objects of the Books R Us application.


User
-firstName : String
-lastName : String
-email : String
-pwdHash : String
-administratorRole : boolean
-address : String
-city : String
-state : String
-zip : String
-creditCardType : String
-creditCardNumber : String
-creditCardExpirationMonth : String
-creditCardExpirationYear : String
+setFirstName(firstName : String)
+getFirstName() : String
+setLastName(lastName : String)
+getLastName() : String
+setEmail(email : String)
+getEmail() : String
+setPwdHash(pwdHash : String)
+getPwdHash() : String
+setAdministratorRole(admin : boolean)
+getAdministratorRole() : boolean
+setAddress(address : String)
+getAddress() : String
+setCity(city : String)
+getCity() : String
+setState(state : String)
+getState() : String
+setZip(zip : String)
+getZip() : String
+setCreditCardType(cardType : String)
+getCreditCardType() : String
+setCreditCardExpirationMonth(month : String)
+getCreditCardExpirationMonth() : String
+setCreditCardExpirationYear(year : String)
+getCreditCardExpirationYear() : String
Product
-productCode : String
-productImageURL : String
-productTitle : String
-productAuthor : String
-productPrice : double
-productRating : int
(stretch feature if time allows)
-productReviews : String
(stretch feature if time allows)
+setProductCode(productCode : String)
+getProductCode() : String
+setProductImageURL(productImageURL : String)
+getProductImageURL() : String
+setProductTitle(productTitle : String)
+getProductTitle() : String
+setProductAuthor(productAuthor : String)
+getProductAuthor() : String
+setProductPrice(productPrice : double)
+getProductPrice() : double
+setProductRating(productRating : int)
(stretch feature if time allows)
+getProductRating() : int
(stretch feature if time allows)
+setProductReviews(productReviews : String)
(stretch feature if time allows)
+getProductReviews() : String
(stretch feature if time allows)
LIneItem
-item : Product
-quantity : int
-subTotal : double
+setProduct(item : Product)
+getProduct() : Product
+setQuantity(quantity : int)
+getQuantity() : int
+setSubTotal(qty : int, price : double)
+getSubTotal() : double
Cart
-items : ArrayList<LineItem>
-subTotal : double
+addItem(item : LineItem)
+removeItem(item : LineItem)
+getItems() : ArrayList<LineItem>)
+setItems(lineItems : ArrayList<LineItem>)
+setSubTotal(lineItems : ArrayList<LineItem>)
+getSubTotal() : double



JSP Outline

The list below shows the graphical user interface pages for the Books R Us application.

- header.jsp – Common header template design used for all pages to ensure uniform theme, includes site menu
- footer.jsp – Common footer template design used for all pages to ensure uniform theme, includes copyright info
- index.jsp – Main page, front page of store
- browse.jsp – All items shown 10 items at a time, summary information with more info link
- product.jsp – Page for a specific product with all information and ability to add to cart
- login.jsp – Allows a user to log in to the site
- register.jsp – Allows a new user to register an account
- registerFail.jsp -Notice user failed to create an account
- registerSuccess.jsp - Notice user create an account successfully
- cart.jsp – Allows a user to view the current contents of their cart and checkout or continue shopping
- checkout.jsp – Page to process the checkout and payment of an order
- confirmation.jsp – Displays after checkout showing that the order has completed

Servlets Outline

The list below shows the controller servlets for the Books R Us application.

- BooksRUsController – Main servlet which processes all requests and forwards them to the sub servlets.
- ProductCatalogServlet – Sub-servlet that is used to manage the displaying of products to the user while browsing the store.

- CartServlet – Sub-servlet that manages the session's cart adding and removing products allowing for quantity changes.
- AccountServlet – Sub-servlet that manages the user's registration and logins/logouts.
- CheckoutServlet – Sub-servlet to process the cart checkout, collecting shipping and billing info and email customer order details upon confirmation.

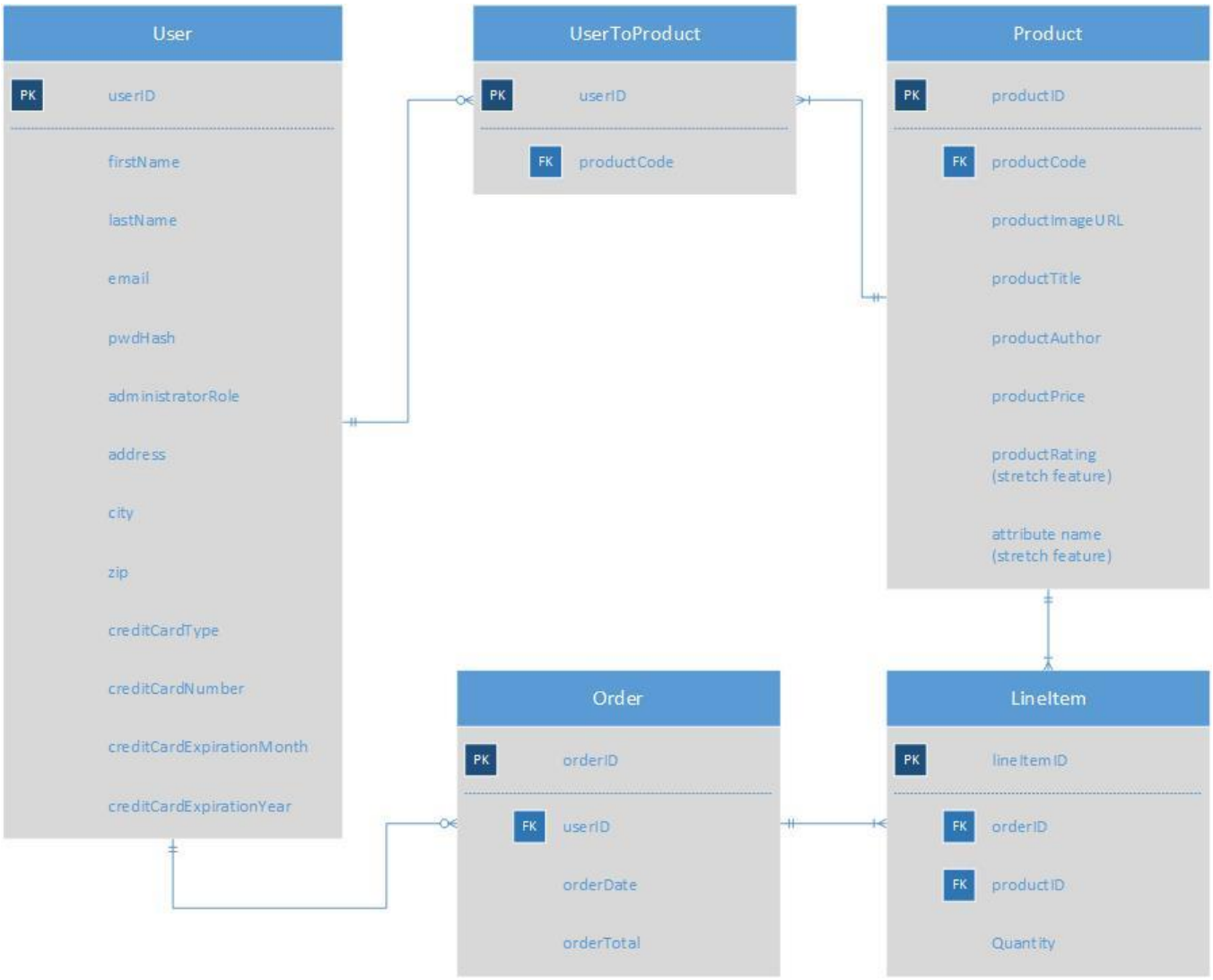
Database and Helper Classes

The list below shows multiple data access and other helper classes for the Books R Us application.

- ConnectionPool - Class that provides a database connection to any class in the application.
- DBUtil - Helper class to close database objects.
- UserDatabase - Class that contains methods to access the User table such as:
    - emailExists to verify if an email is in the database User table
    - insert to add a user record to the database
    - update to update a user record in the database
    - delete to delete a user record from the database
    - select to select a user record from the database
- ProductDatabase - Class that contains methods to access the Product table such as:
    - selectProduct to return a product record from the database
    - selectProducts to return a set of product records from the database (e.g. ArrayList<Product>)
- OrderDatabase - Class that contains methods to access the Order table such as:
    - insert to add a line item in the Order table
    - update to modify an existing order record in the database
    - select to return an order record
- LineItemDatabase - Class that contains methods to access the LineItem table such as:
    - insert to add a line item in the LineItem table
    - selectLineItems to return from the database a set of line items for a particular order
- UserToProductDatabase - Class that contains methods to access the UserToProduct table such as:
    - insert to add a line item in the UserToProduct table
- mailUtil - Helper class to send a message. It may be used by the CheckoutServlet to send an email to the customer with order details. This class gets an email session, creates the message, addresses the message, and sends the message.

Database Diagram

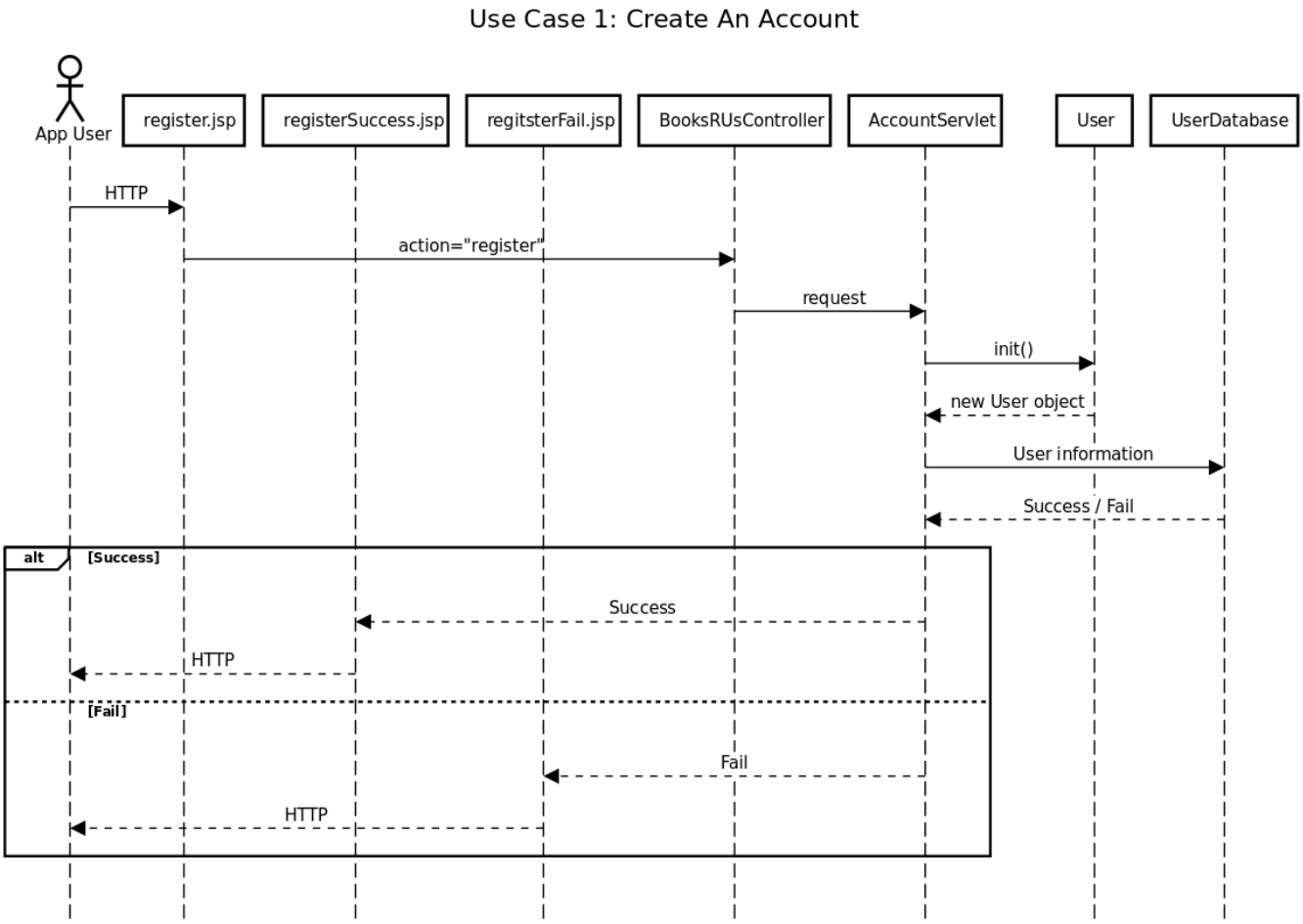
The database diagram below shows the database tables where the application will store its data.



Sequence Diagrams

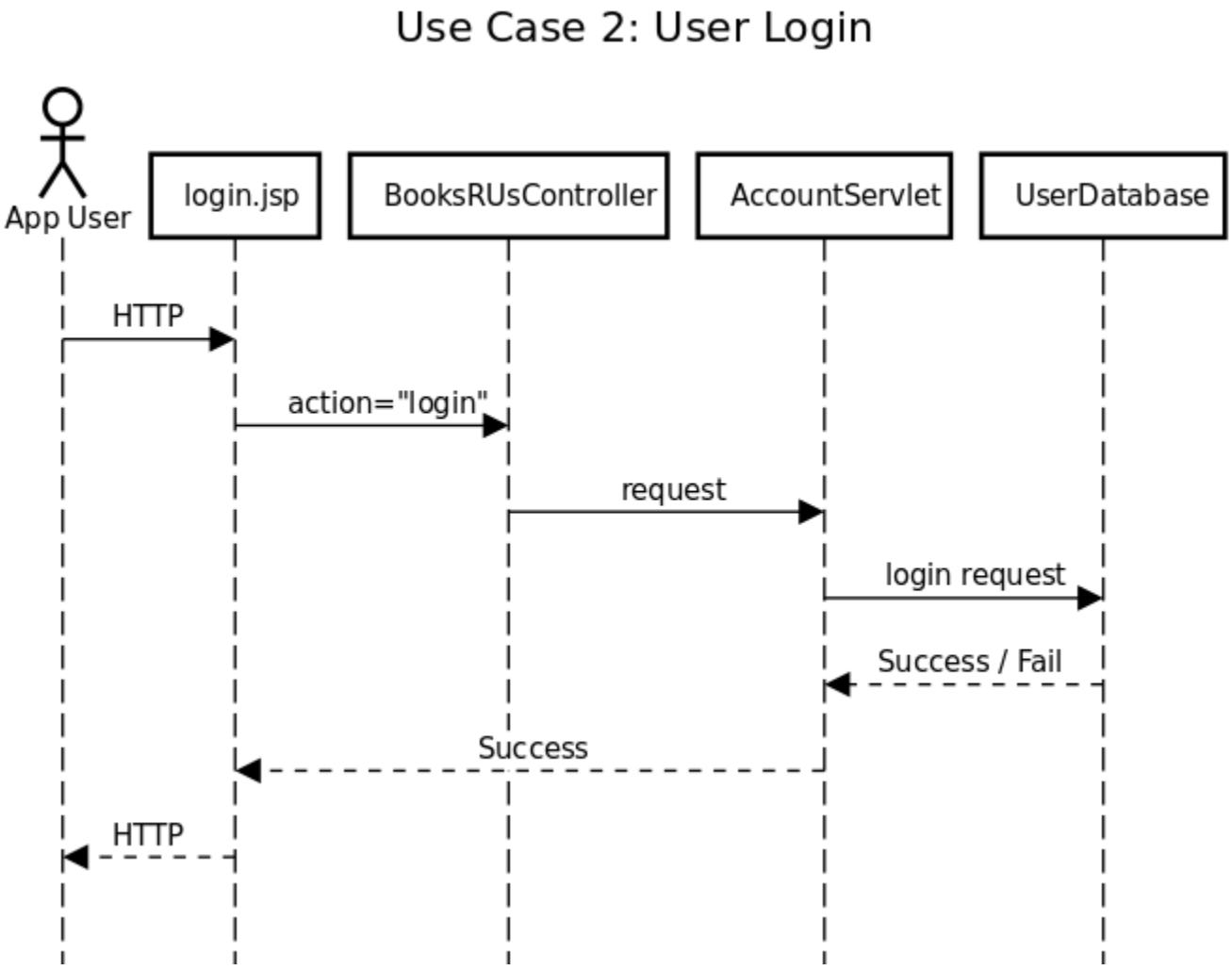
Use Case 1: Create an account

This is a preliminary sequence diagram that will be updated as we review our design and get our implementation in place.



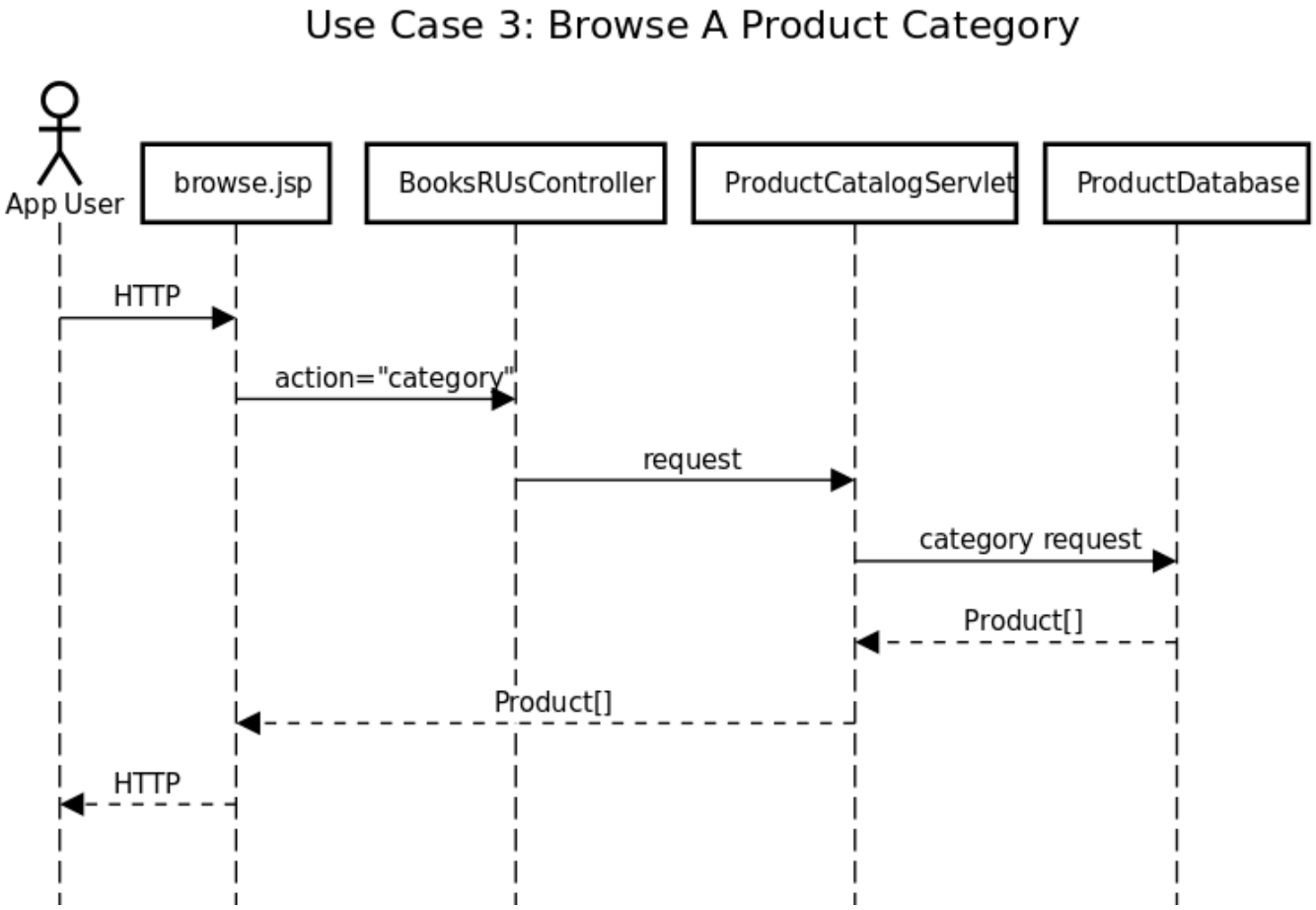
Use Case 2: User login

This is a preliminary sequence diagram that will be updated as we review our design and get our implementation in place.



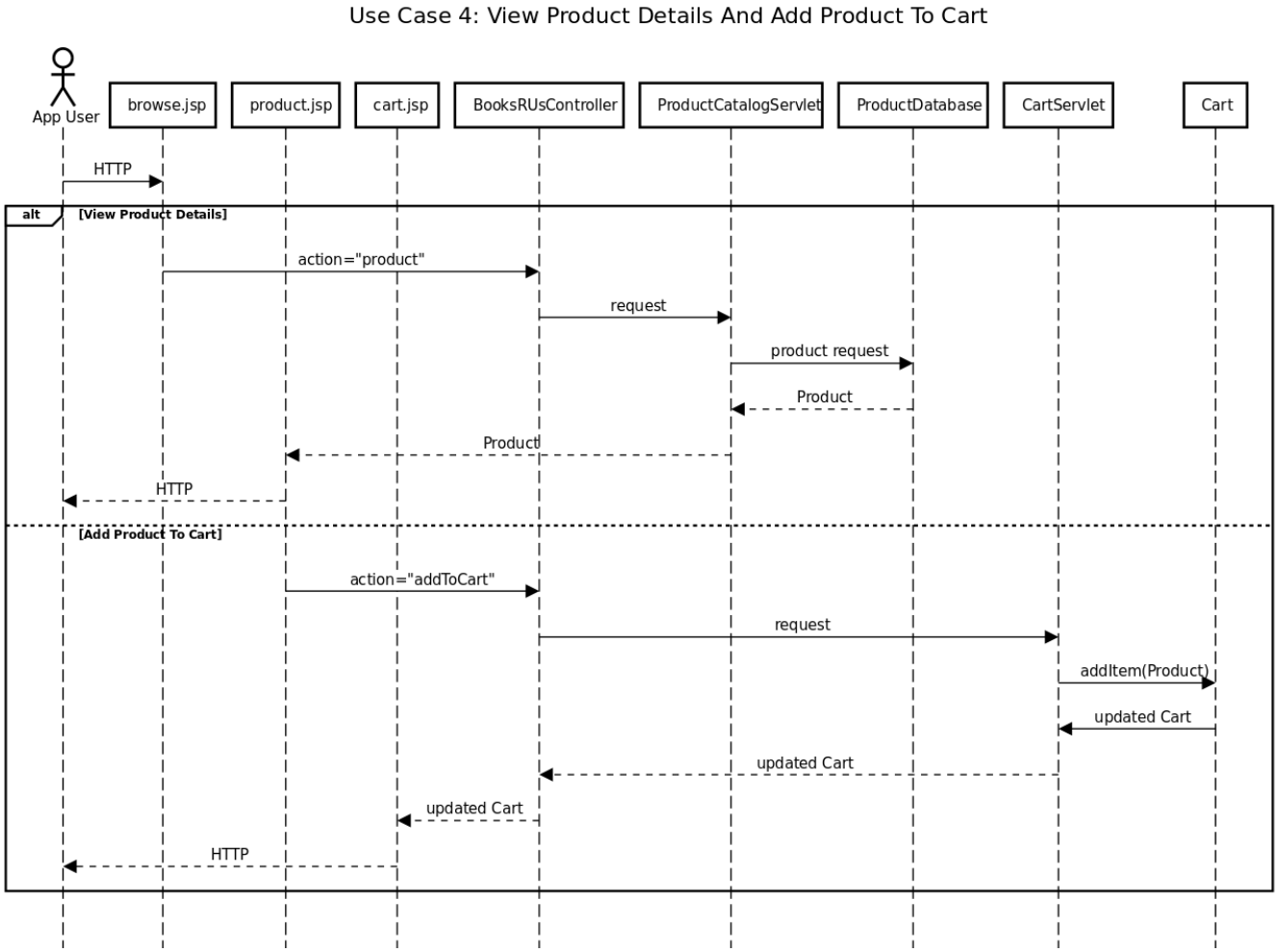
Use Case 3: Browse a product category

This is a preliminary sequence diagram that will be updated as we review our design and get our implementation in place.



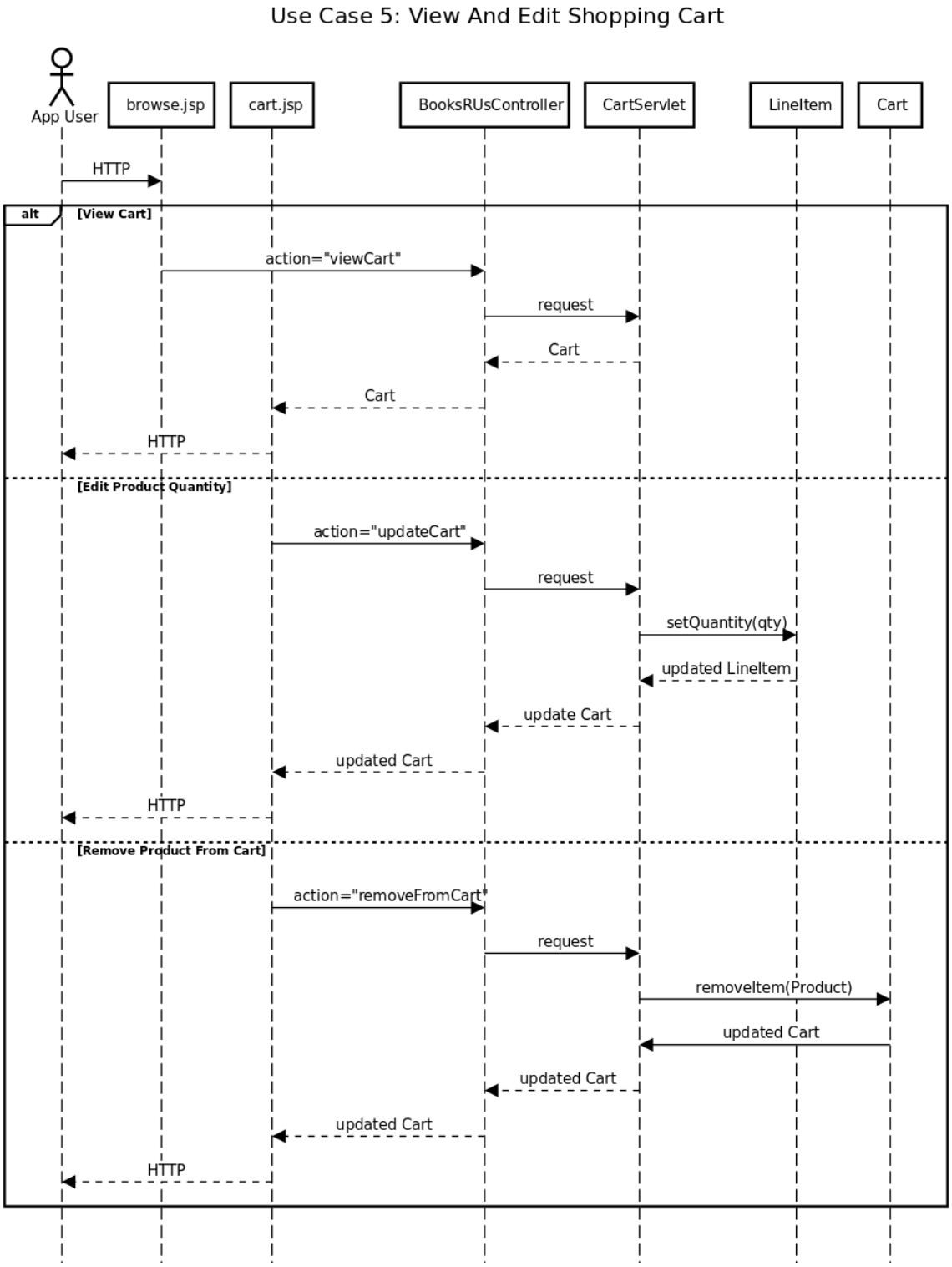
Use Case 4: View product details and add product to cart

This is a preliminary sequence diagram that will be updated as we review our design and get our implementation in place.



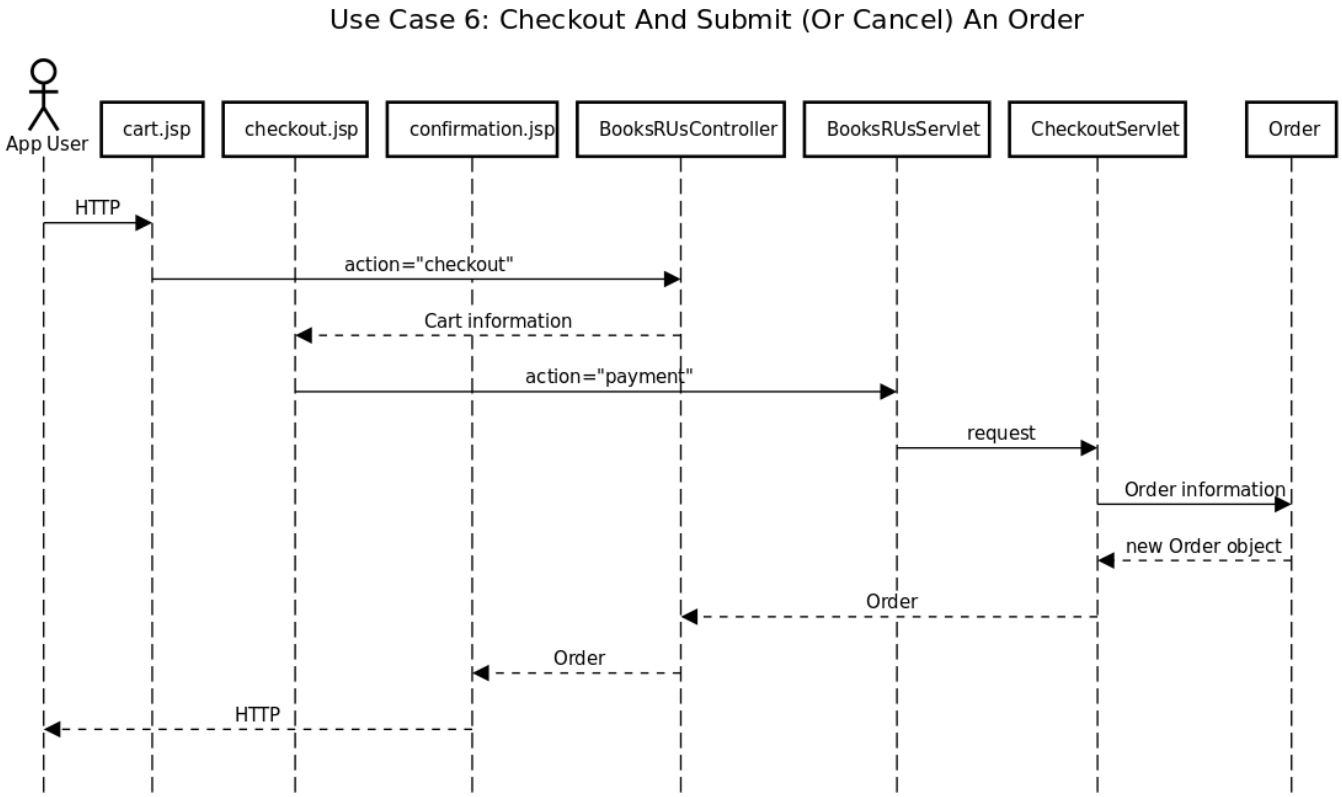
Use Case 5: View and edit shopping cart

This is a preliminary sequence diagram that will be updated as we review our design and get our implementation in place.



Use Case 6: Checkout and submit (or cancel) an order

This is a preliminary sequence diagram that will be updated as we review our design and get our implementation in place.

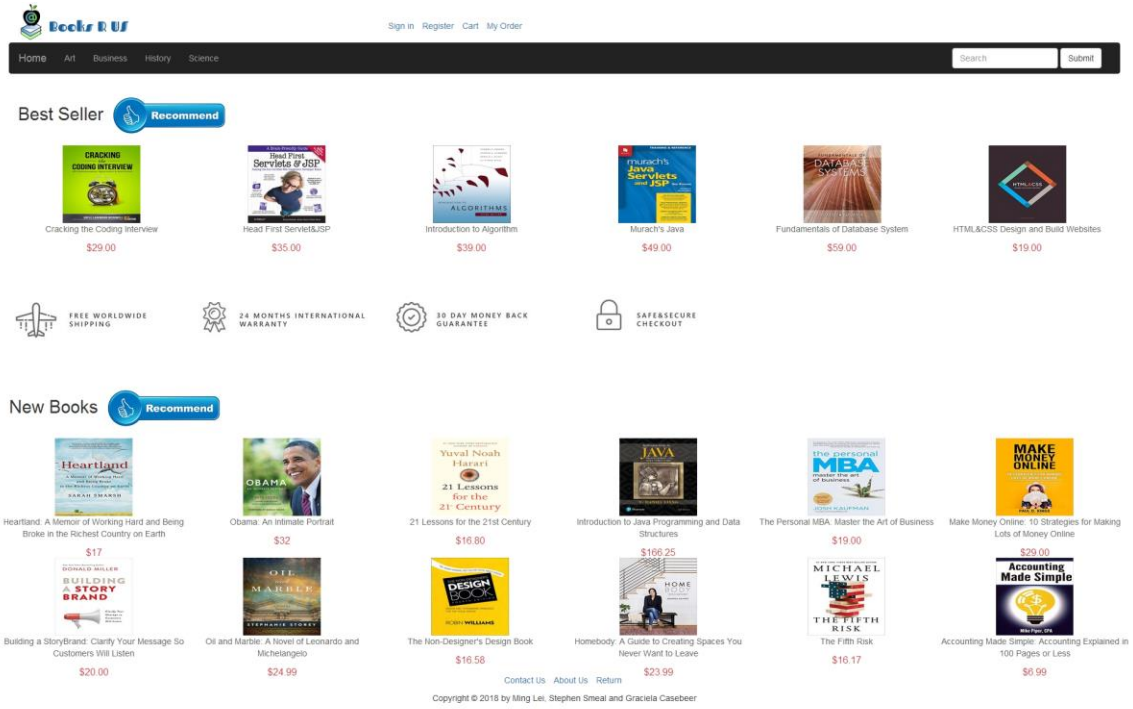


Use Case 7: Edit product profile (stretch feature)

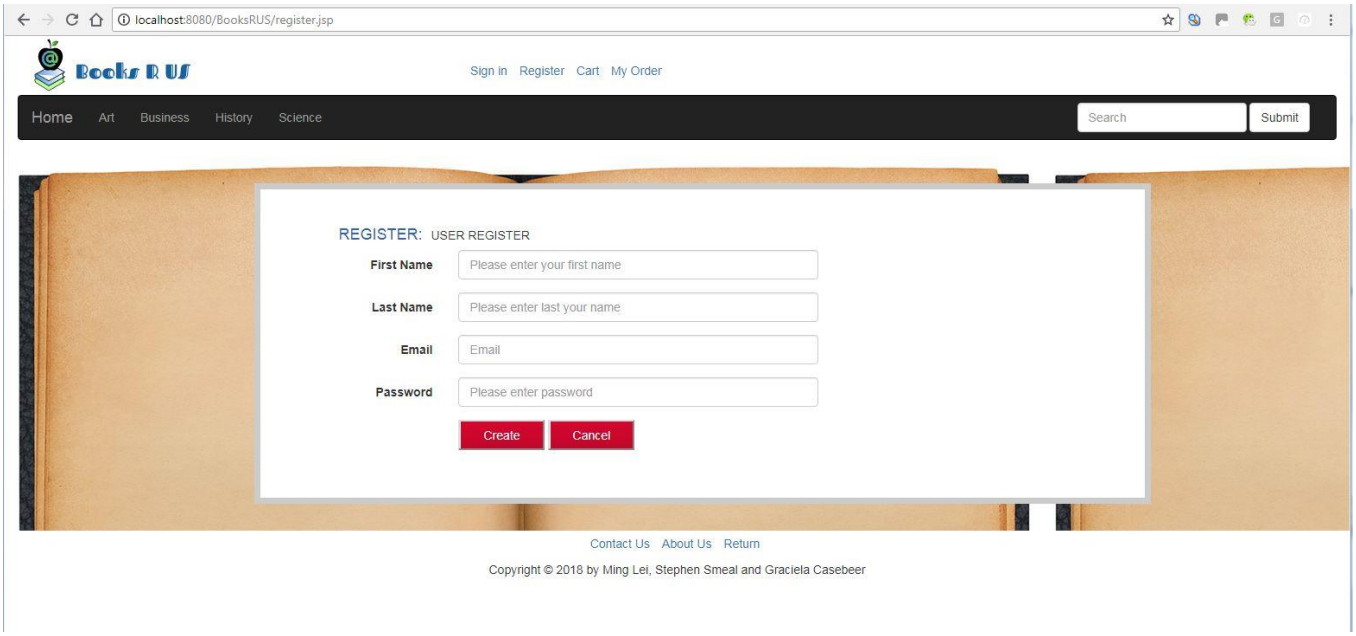
<Use Case 7 Diagram (to be designed for stretch feature)>

Appendix A - Screenshot Samples

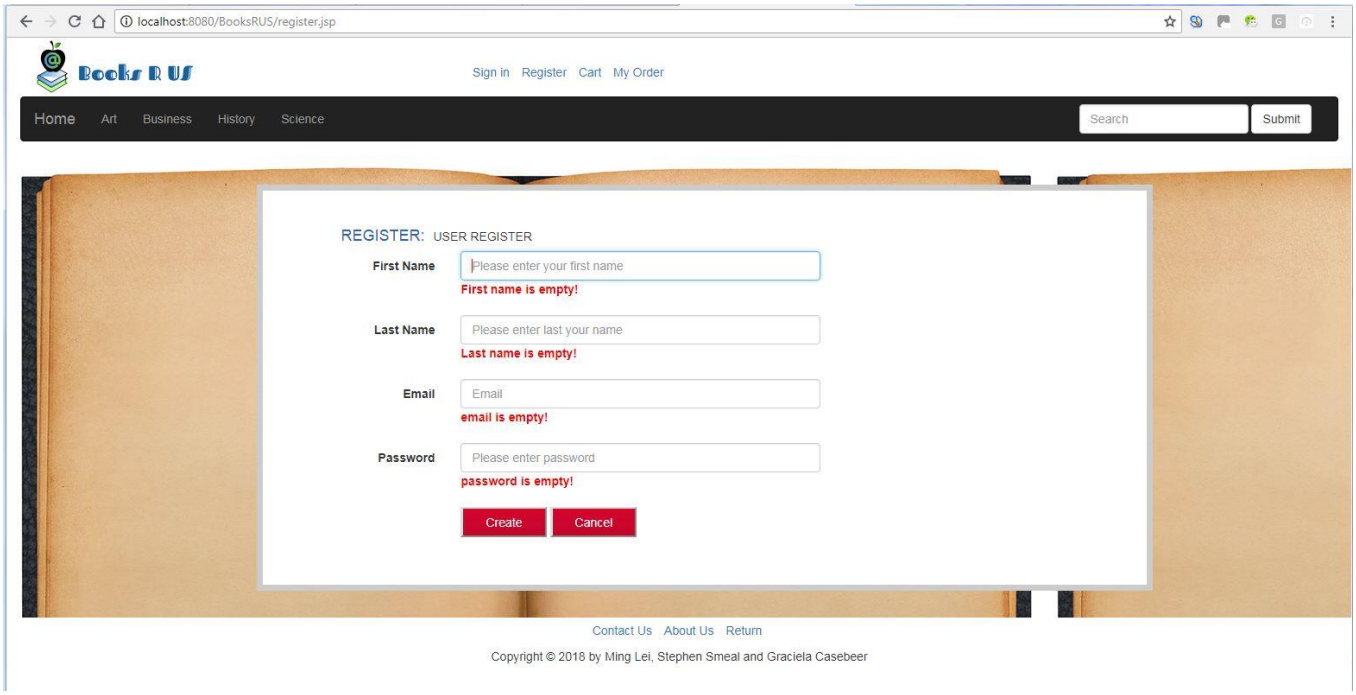
Main page (index)



Registration (register)



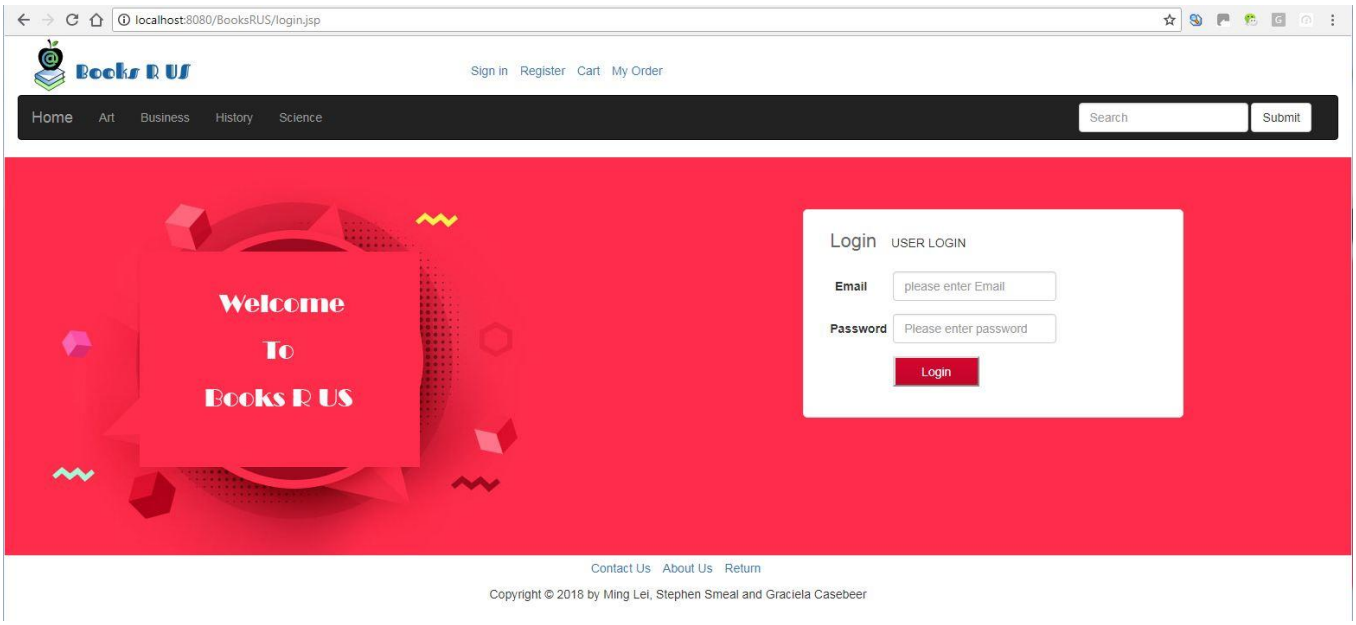
Registration Validation



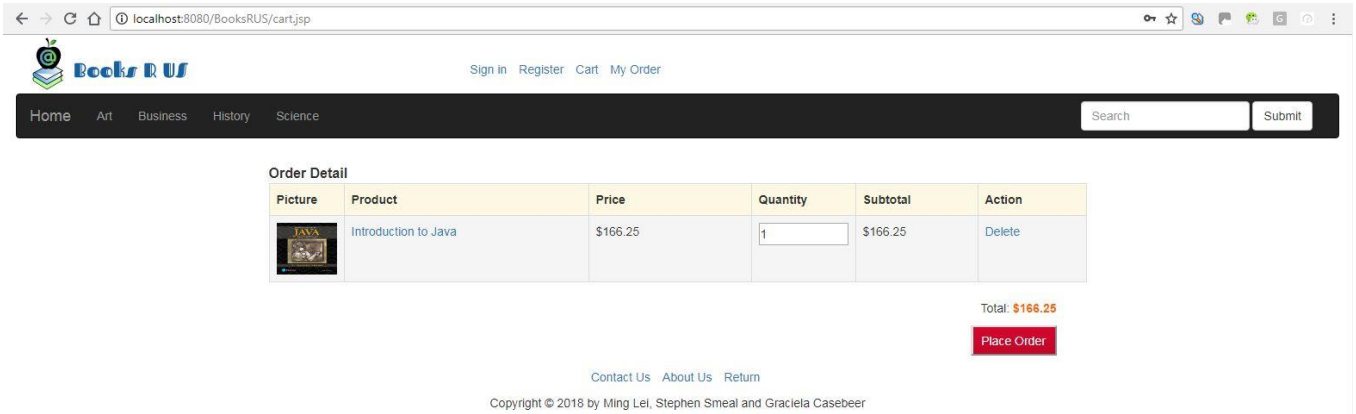
Registration Success

Registration Failed

Sign In/Login


localhost:8080/BooksRUS/login.jsp
Books R US
Sign in Register Cart My Order
Home Art Business History Science
Search
Submit
Welcome To Books R US
Login USER LOGIN
Email
please enter Email
Password
Please enter password
Login
Contact Us About Us Return
Copyright © 2018 by Ming Lei, Stephen Smeal and Graciela Casebeer

Cart


localhost:8080/BooksRUS/cart.jsp
Books R US
Sign in Register Cart My Order
Home Art Business History Science
Search
Submit
Order Detail
Picture
Product
Price
Quantity
Subtotal
Action
Introduction to Java
$166.25
1
$166.25
Delete
Total: $166.25
Place Order
Contact Us About Us Return
Copyright © 2018 by Ming Lei, Stephen Smeal and Graciela Casebeer

Product Information