Derive Use Cases Thinking About the Actors

Think about the different actors involved interacting with the system:

1. Customer
2. Shipping Clerk
3. Seller
4. Customer Service
5. Webmaster

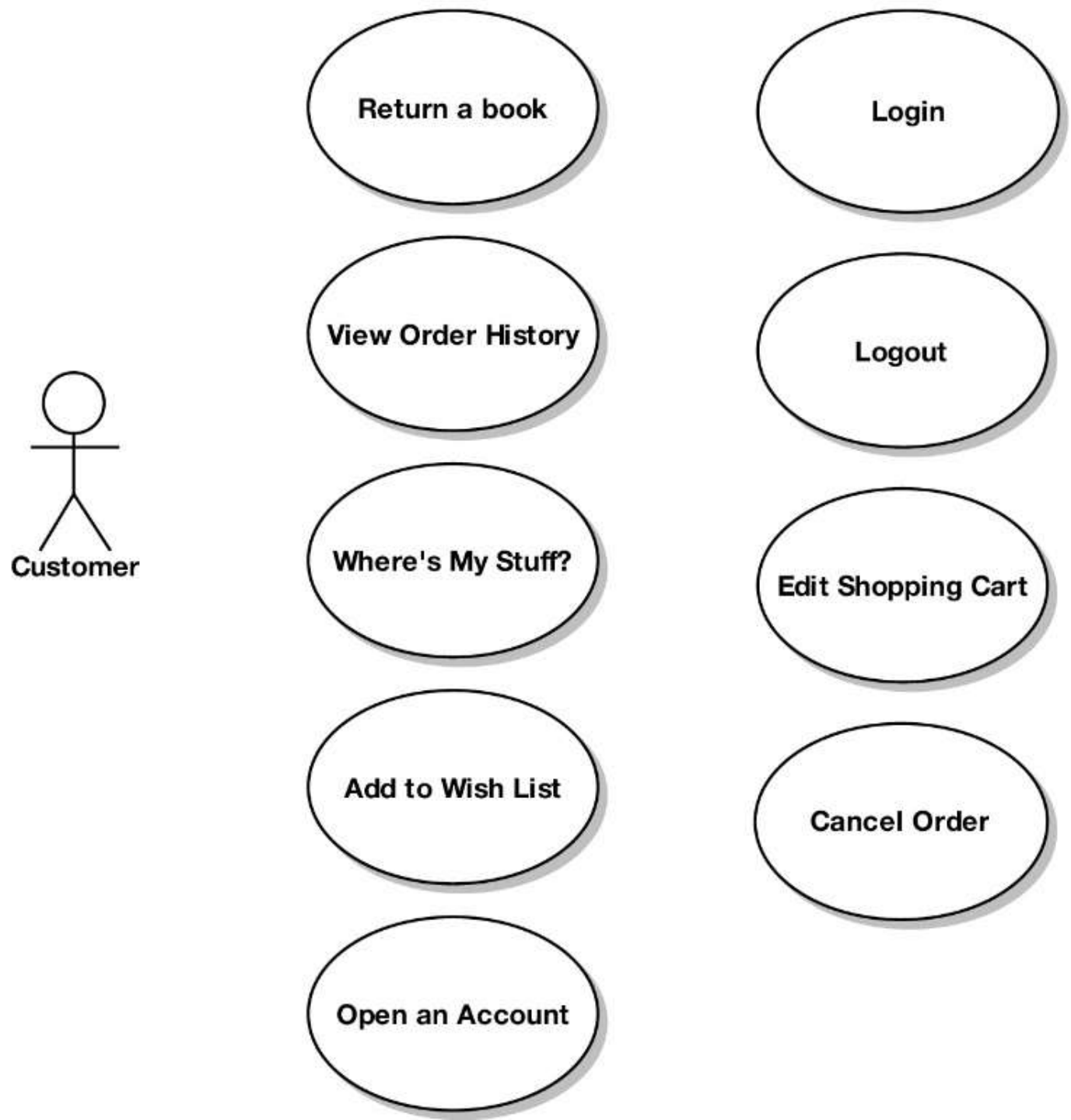
Customer Related Requirements

- The bookstore must be able to sell **books**, with **orders** accepted over the **Internet.**
- The user must be able to add books into an online **shopping cart**, prior to **checkout**.
  - Similarly, the user must be able to remove **items** from the shopping cart.
- The user must be able to maintain **wish lists** of books that he or she wants to purchase later.
- The user must be able to cancel orders before they've shipped.

Customer Related Requirements (Cont.)

- The user must be able to pay by **credit card** or **purchase order**.
- The user must be able to create a **customer account**, so that the system remembers the user's details (name, address, credit card details) at login.
  - The system shall maintain a **list of accounts** in its central database.
  - When a user logs in, his or her **password** must always be matched against the passwords in the **master account list**.

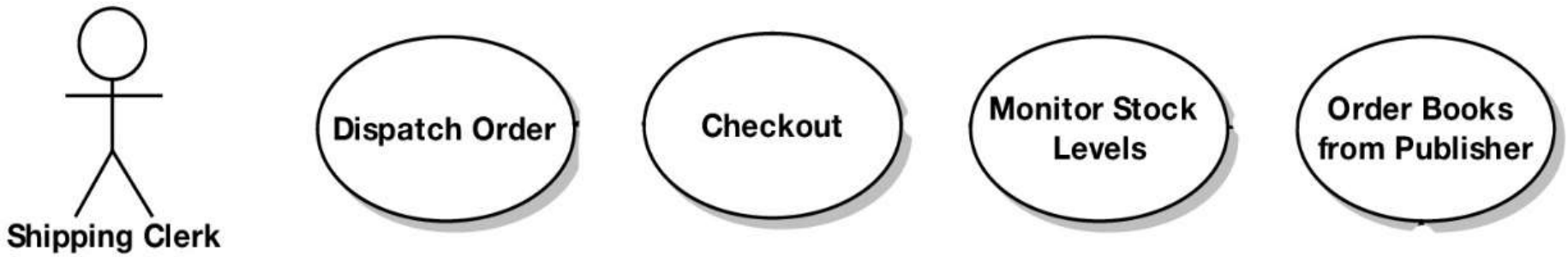
Customer Interaction Use Cases



Shipping Clerk Related Requirements

- The **shipping fulfillment system** shall be carried out via Amazon Web Services.

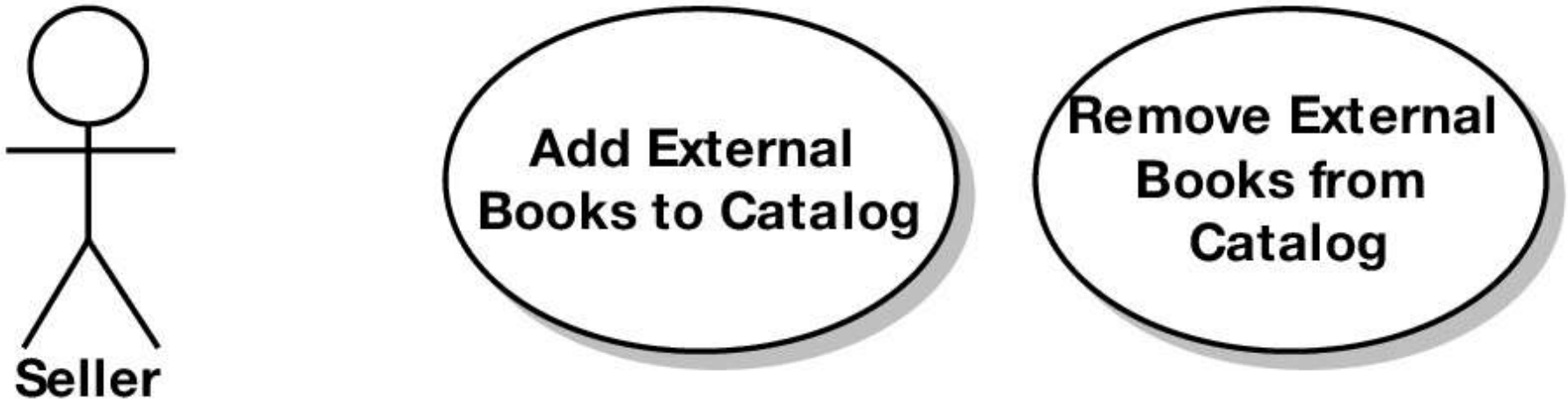
Shipping Clerk Interaction Use Cases



Seller Related Requirements

- The bookstore shall allow third-party **sellers** (e.g., second-hand bookstores) to add their own individual **book catalogs**. These are added into the overall **master book catalog** so that sellers' books are included in search results.

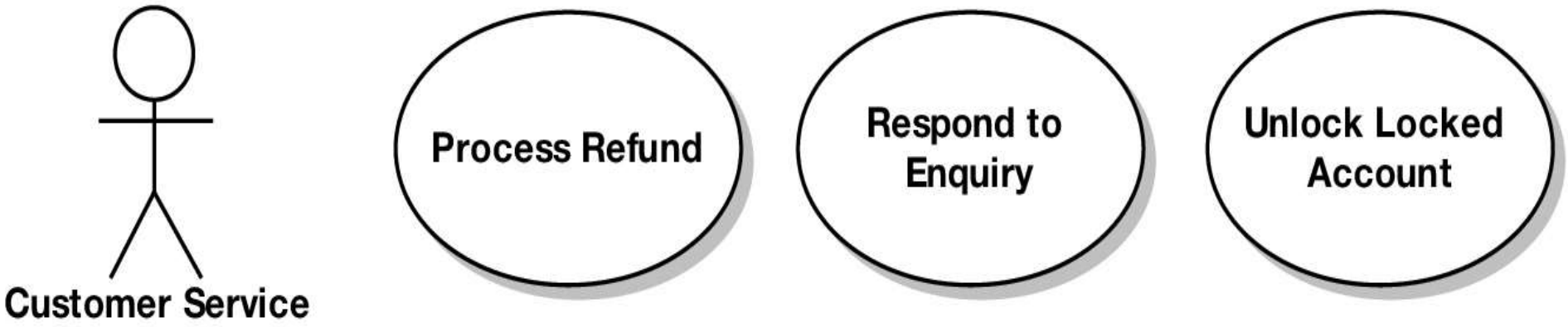
Seller Interaction Use Cases



Customer Service Related Requirements

- The user must be able to create a **customer account**, so that the system remembers the user's details (name, address, credit card details) at login.
  - The system shall maintain a **list of accounts** in its central database.
  - When a user logs in, his or her **password** must always be matched against the passwords in the **master account list**.
- The user must be able to cancel orders before they've shipped.

Customer Service Interaction Use Cases



Webmaster Related Requirements

- The bookstore must be embeddable into **associate partners**' websites using **minicatalogs**, which are derived from an overall **master catalog** stored in a central **database**.
- **Book reviews** must be moderated—that is, checked and "OK'd" by a member of staff before they're published on the website.
- It must be possible for staff to post **editorial reviews** of books. These should also appear on the book details screen.
- The bookstore shall allow third-party **sellers** (e.g., second-hand bookstores) to add their own individual **book catalogs**. These are added into the overall **master book catalog** so that sellers' books are included in search results.

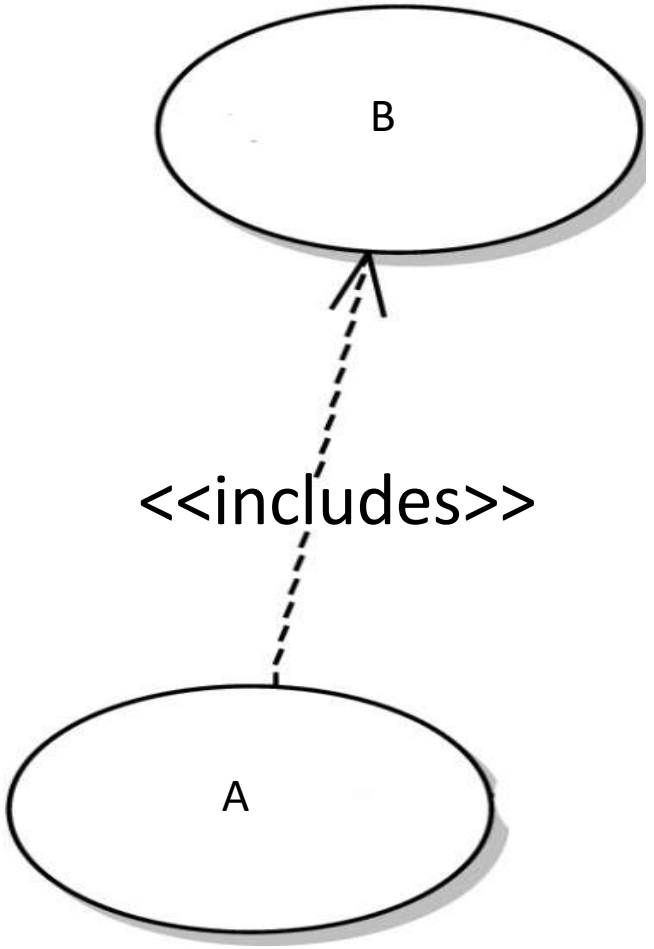
Webmaster Interaction Use Cases

Use Case Simple Relationships <<includes>>

A <<includes>> B

Halfway through use case A, use case B is called. When B finishes, A carries on from where it left off.

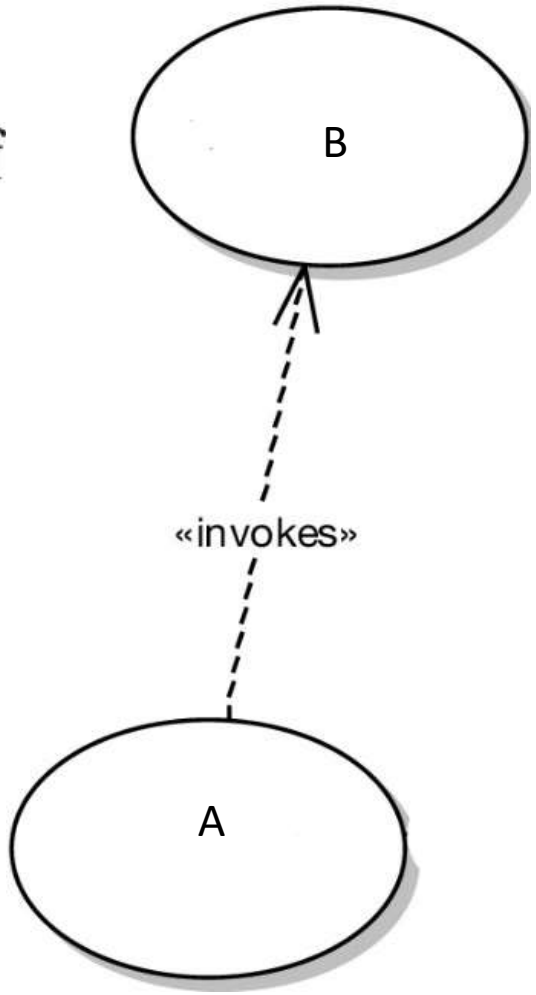
- We use <<invokes>> which is a superset of <<includes>>.
- Notice that <<includes>> can also be used as <<include>> also.



Use Case Simple Relationships <<invokes>>

A <<invokes>> B

Use case B happens **during** the lifespan of use case A.

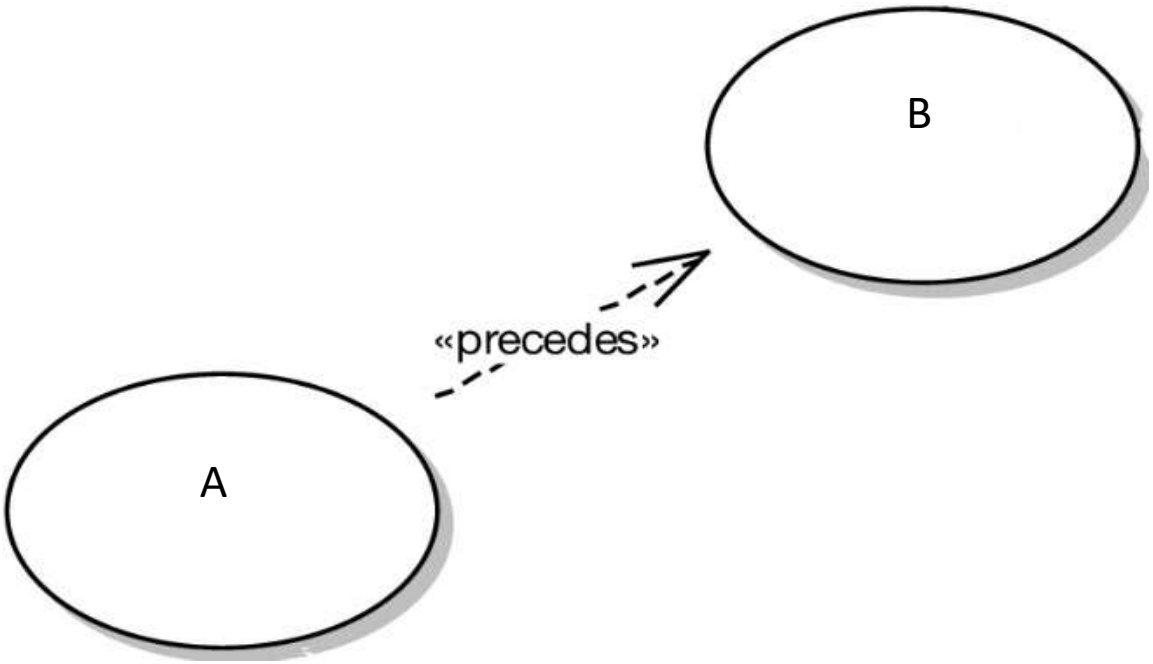


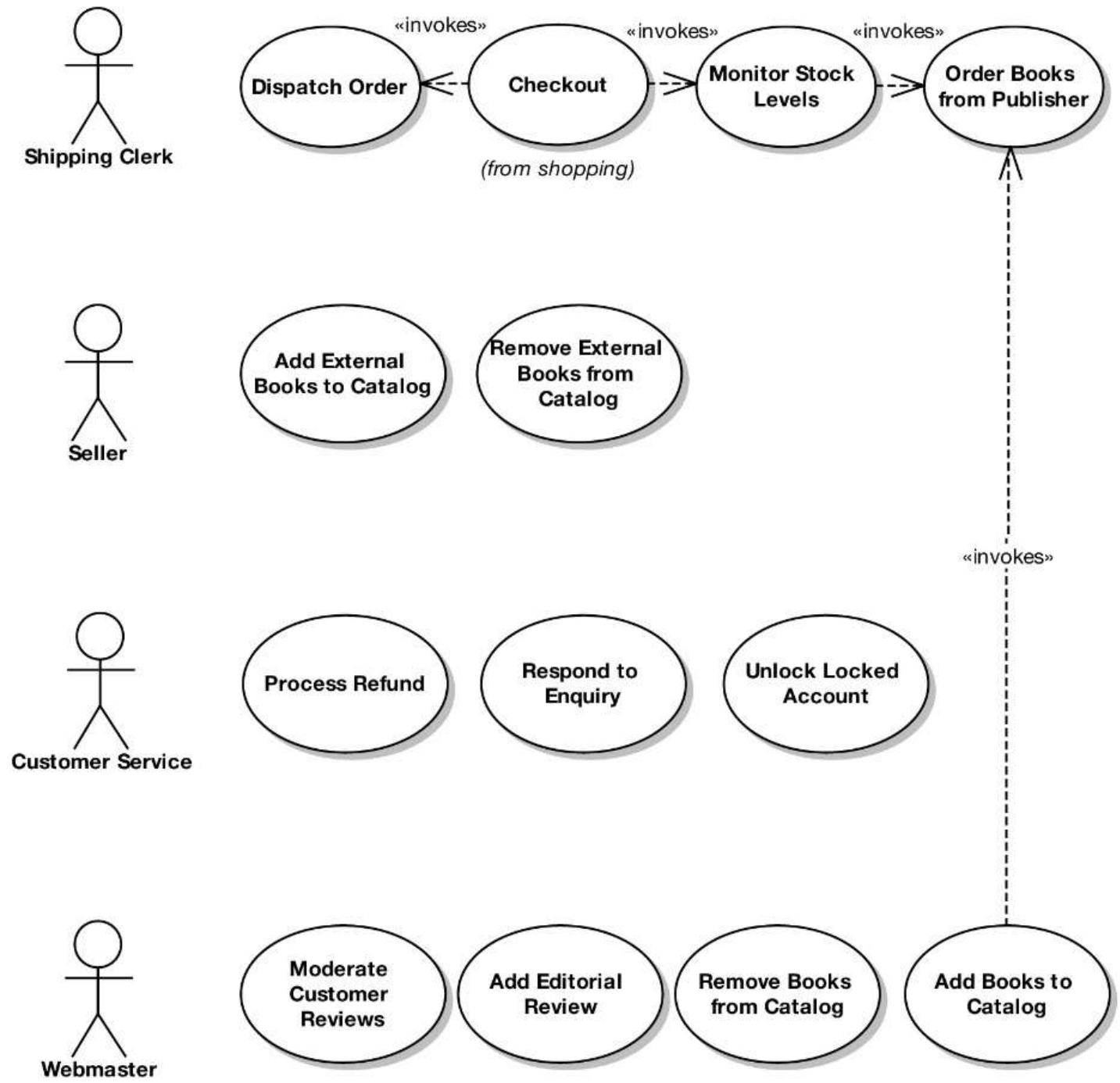
- We use <<invokes>> which is more general than <<includes>>.
- <<invokes>> is a superset of <<includes>>.

Use Case Simple Relationships <<precedes>>

A <<precedes>> B

Use case A must take place in its entirety **before** use case B even begins.



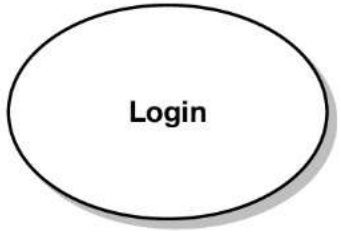


Some Relationships Added

More Relationships Added

- You add relationships as you do your analysis, but keep in mind that the textual description is by far the most important part of use cases.

Diagram



Write Use Cases in Active Voice

The wrong way to write use case descriptions; passive voice (don't tell them):

The capability is provided for users to log in, using a password-protected authorization scheme.

The right way to write use case descriptions; active voice (show them):

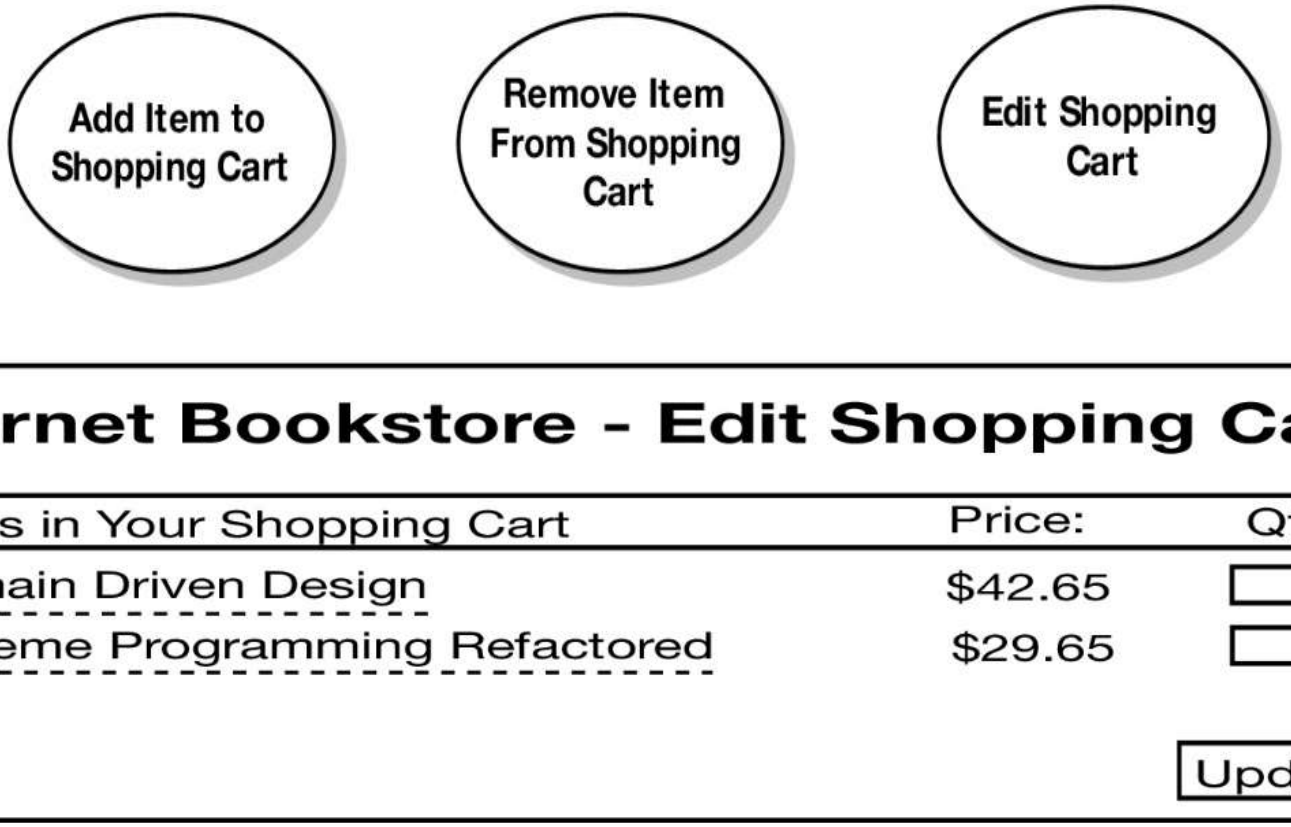
The user enters her username and password, and then clicks the Login button. The system looks up the user profile using the username and checks the password. The system then logs in the user.

An Even Better Way

The system displays the Login screen. *The user enters her username and password, and then clicks the Login button. The system looks up the user profile using the username and checks the password. The system then logs in the user.*

- Even better is to mention the screen the user uses by name to interact with the system. The Login screen will represent a boundary class later on.
- The idea of use case textual descriptions is to demonstrate (show) the behavior through user-system or system-user interactions, rather then to tell them.
- Also, showing screens with information implies initialization.

Use Visual Aids Like Screen Mock-ups



- This screen is a boundary class, and it will help in writing how the customer actor interacts with the system.

Description Using the *Edit Shopping Cart* Screen Mock-up

The user clicks the Edit Shopping Cart button, and the system shows the Edit Shopping Cart page with a list of books in the user's shopping cart. The user selects one of the books, changes the quantity, and clicks the Update button. The system shows the page with the quantities and price totals updated.

- Be sure you name each screen; "Edit Shopping Cart" in this case. These are boundary objects.
- Think about the use case text as text you could find in a user's guide.

Reference Domain Classes by Name

- The domain model includes the following domain classes:
  - Wish List
  - Book
  - Book List
  - Shopping Cart

Reference Domain Classes by Name (Cont.)

Not using the names of the classes of the domain model:

The user selects a title *and adds it to his* list of books to be saved for later. *The system displays a page with the updated list and also shows a* list of titles in the user's cart, ready for checkout.

Using the names of the classes of the domain model:

The user selects a ***Book*** *and adds it to his* ***Wish List****. The system displays a page with the updated list and also displays the user's* ***Shopping Cart****.*

- We still have a problem with this text; what is it?
- Name the screens; for example "Wish List Page".

Progressive Improvement of Use Case Text

The user types in a review of the selected item, gives it a score, and sends it. The review is sent to a moderator.

- The user interactions with the system and the system responses are not captured in this text description
- The description is not using the domain classes from the domain model

Progressive Improvement of Use Case Text (Cont.)

The Customer selects the book. The system displays a book detail page. The Customer clicks the Write Review button, and the system shows the Write Review screen. The user types in a review of the book, gives it a rating out of five stars, and clicks the Send button. The system ensures that the review isn't too long or short, and that the rating is within one and five stars. The system then displays a confirmation screen, and the review is sent to a moderator, ready to be added.

- The description is going out of the bounds of the use case "Write Reader Review".
- Notice how we have identified a new actor, "Moderator".

Progressive Improvement of Use Case Text (Cont.)

The Customer clicks the Write Review button for the book currently being viewed, and the system shows the Write Review screen. The Customer types in a Book Review, gives it a Book Rating out of five stars, and clicks the Send button. The system ensures that the Book Review isn't too long or short, and that the Book Rating is within one and five stars. The system then displays a confirmation screen, and the review is sent to a Moderator, ready to be added.

- Here the description is within the bounds of the use case. Also, notice how it is using the terms of the domain model such as "Book Review" and "Book Rating".

Progressive Improvement of Use Case Text (Cont.)

- Notice that in the domain model we have defined an actor called customer not reader. The use case title should change to match with the domain model; from “Write Reader Review” to “Write Customer Review”.

Progressive Improvement of Use Case Text (Cont.)

- We have not yet thought about what happens if the customer does something wrong; that would be part of the alternate path of the textual description.
- In addition to the customer doing something wrong, think about what unusual actions the user could take with respect to the use case “Write Customer Review”.

Progressive Improvement of Use Case Text (Alternate Path)

- What happens if the review is too short?
- What happens if the customer gives the book a rating of 50?
- What happens if the customer gives a book review that is too long?
- What happens if the customer is not logged in?

Progressive Improvement of Use Case Text (Alternate Path) (Cont.)

ALTERNATE COURSES:

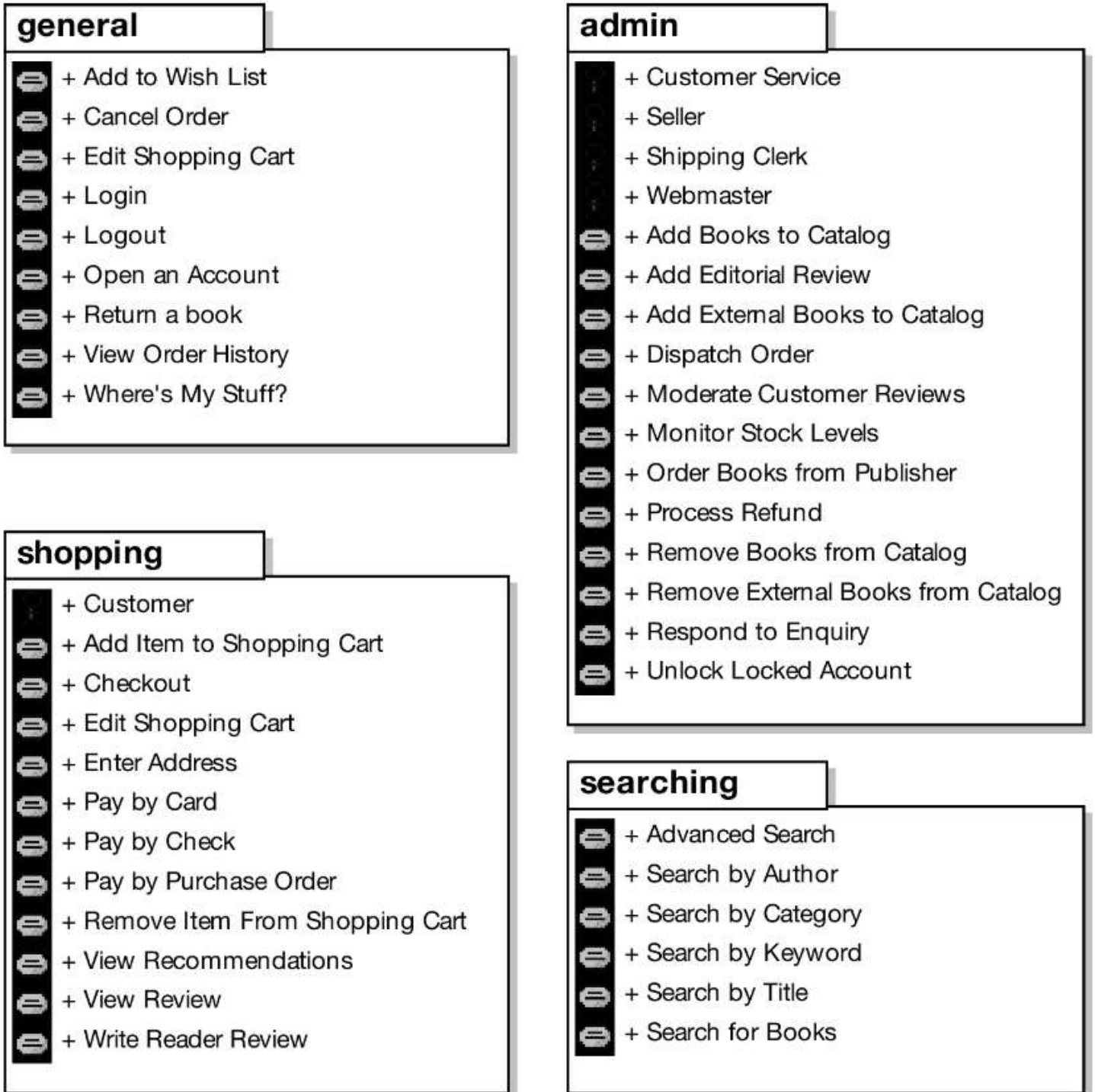
User not logged in: *The user is first taken to the Login screen and then to the Write Review screen once he is logged in.*

The user enters a review that is too long (text > 1MB): *The system rejects the review and responds with a message explaining why the review was rejected.*

The review is too short (< 10 characters): *The system rejects the review.*

Is It a Use Case or Algorithm?

- An algorithm is also a sequence of steps.
- The difference is that an algorithm does not have sequential interactions with actors.
- From a use case perspective, an algorithm represents a single step in a use case.
- The algorithm should be given a name in the use case behavior sequence. For example, “Sort List”, “Give Recommendations”, etc.



Organize Use Cases Into Packages

- Use cases should be grouped in logically in terms of functionality subsystems.

“Search by Author”

BASIC COURSE:

The system displays the page with the search form; the user clicks the Author field and types in an author name (e.g., Fred Smith). The user clicks the Search button; the system reads the search form, looks up any books matching that author name, and displays them in a list.

ALTERNATE COURSES:

No matching books found: A page is displayed informing the user that no matching books were found.

Improved “Search by author”

The same problem can be found several times in this use case: the boundary objects haven’t been given explicit names. The fixed version follows.

BASIC COURSE:

The system displays the Search Page; the user clicks the Author field and types in an author name (e.g., Fred Smith). The user clicks the Search button; the system reads the search form, looks up any books matching that author name, and displays the Search Results page showing the resulting Book List.

ALTERNATE COURSES:

No matching books found: The Search Not Found page is displayed.

- Always name your screens.

“Edit Shopping Cart”

PRECONDITIONS:

The user has logged in.

The user has navigated to the Edit Shopping Cart page.

BASIC COURSE:

The user adds or removes whatever items he wants to change, and then clicks the Update button. The system adds or removes the items, and then displays the page with the updated shopping cart.

ALTERNATE COURSES:

Shopping cart is empty: No items can be removed.

“Edit Shopping Cart” Problems

- In general to vague and ambiguous.

There are at least three problems with this use case.

Problem 1: The use case includes a “Preconditions” clause. Although on very rare occasions, you might find that it’s useful to include this clause, most of the time it serves no appreciable purpose. In this example, it actually throws the use case text off course, as the initial “display” action is missed. This would in turn be missed out on the robustness diagram, meaning it would likely be skipped on the design, not estimated for, and not tested.

Problem 2: The basic course text is a bit woolly. It doesn’t describe a specific scenario, but instead tries to cover all bases (“The user adds or removes whatever item . . .”). As a result, an important behavioral aspect is missed: the user wouldn’t necessarily want to add items from this page, just remove them (or change the quantity).

Problem 3: The alternate course doesn’t tie into any particular action in the use case text. There are also several relatively obvious alternate courses that are missing.

Improved "Edit Shopping Cart"

BASIC COURSE:

The system displays the Shopping Cart page. The user clicks the Remove button next to a Line Item. The system removes the item from the user's Shopping Cart, and then redisplays the page. The user then clicks the Quantity text field for another Line Item, changes its value from 1 to 2, and clicks the Update button. The system updates the Shopping Cart, recalculates the total amount, and redisplays the page.

ALTERNATE COURSES:

Item not found: The item that the user chose to remove wasn't found in the Shopping Cart (this could happen if the user had two browser tabs open and is viewing an older version of the page). The system refreshes the Shopping Cart page, along with a warning message that the user's action failed because the page was out of date.

Quantity changed to zero: This counts as removing the item, so the item is removed from the Shopping Cart.

Negative value or non-numeric "value" entered: The page is redisplayed with the original Quantity value, and a message next to it informs the user that he entered an invalid value.

“Open an Account”

BASIC COURSE:

The system displays the Create New Account page and enters the following fields: Username (must be unique), password, confirm password, first name, last name, address (first line), address (second line), city, state, country, zip/postal code, telephone number, and e-mail address. Then the user clicks the Submit button; the system checks that the Username is unique, creates the new user account, and displays the main Hub Page, along with a message indicating that the user account is now created and logged in.

ALTERNATE COURSES:

Password and Confirm Password don’t match: The page is redisplayed with a validation message.

Username not unique: The page is redisplayed and the user is asked to choose a different username.

Improved "Open an Account"

- Don't get hung up on presentation details in your descriptions. Update the Customer domain class with those attributes.

BASIC COURSE:

The system displays the Create New Account page and enters the fields to define a new Customer account (user-name, password, address, etc.). Then the user clicks the Submit button; the system checks that the Username is unique, creates the new user account, and displays the main Hub Page, along with a message indicating that the user account is now created and logged in.

ALTERNATE COURSES:

Password and Confirm Password don't match: The page is redisplayed with a validation message.

Username not unique: The page is redisplayed and the user is asked to choose a different username.

”Show Book Details”

***USE CASE**: Show Book Details*

***LEVEL**: User Goal*

***PRECONDITIONS**:*

1. *The user MUST be viewing the website.*

2. *The user MAY be logged in.*

”Show Book Details” (Cont.)

BASIC COURSE:

If the user has an account, he MAY log in first (though this isn't essential for this use case). The user MAY navigate to the website's main area. This area MUST be easily accessible and SHOULD provide easy access to a search area and various book directories (e.g., top 10 bestsellers in different categories, editors' picks, etc.). Then the user browses the details for a book title. The system displays the screen showing the book information. If the user wishes to, he may proceed to purchase the book. Allowed payment options are as follows: Visa, American Express, and check (the check must be received by the billing department before the order is dispatched).

- We are only concerned in the steps involved in viewing book details, so the steps of proceeding to purchase the book is out of scope.

”Show Book Details” (Cont.)

POSTCONDITIONS:

1. *The Book Details screen is being shown.*

2. *The details for the selected book have been retrieved.*

”Show Book Details” (Cont.)

- Be sure to begin the use case with behavior that is the beginning of that use case, not the behavior of another use case.
- The preconditions and postconditions are outside the scope of this use case.
- Our goal is to link the use cases to classes.
- If you feel the preconditions help, transfer them to a diagram and use the <<precedes>> link between the two use cases.

Diagram

”Show Book Details” Improvement-1

BASIC COURSE:

If the user has an account, he MAY log in first (though this isn't essential for this use case). The user MAY navigate to the website's main area. This area MUST be easily accessible and SHOULD provide easy access to a search area and various book directories (e.g., top 10 bestsellers in different categories, editors' picks, etc.). Then the user browses the details for a book title. The system displays the screen showing the book information. If the user wishes to, he may proceed to purchase the book. Allowed payment options are as follows: Visa, American Express, and check (the check must be received by the billing department before the order is dispatched).

- We got rid of the preconditions.

”Show Book Details” Improvement-2

BASIC COURSE:

The user MAY navigate to the website's main area. This area MUST be easily accessible and SHOULD provide easy access to a search area and various book directories (e.g., top 10 bestsellers in different categories, editors' picks, etc.). Then the user browses the details for a book title. The system displays the screen showing the book information.

- We are only concerned in the steps involved in viewing book details, so the steps of proceeding to purchase the book is out of scope.

”Show Book Details” Improvement-3

BASIC COURSE:

The Customer MAY navigate to the Bookstore's main area. This area MUST be easily accessible and SHOULD provide easy access to a search area and various Bestseller Lists. Then the Customer browses the details for a book title. The system displays the screen showing the Book Details.

- Use the name of domain classes; in this case instead of **book information** use **“Book Details” .**

”Show Book Details” Improvement-3 (cont.)

- There are still problems; for instance what is the bookstore main area?
- The description is too vague and abstract, it is using passive voice.
- The text should not sound like a requirements document.

”Show Book Details” Improvement-4

ALTERNATE COURSES:

Book not found: *The system displays a Book Details Not Found screen.*

- We need to add the alternate courses.

”Show Book Details” Improvement-5

BASIC COURSE:

The Customer types in the URL for the bookstore's home page, which the system displays. Then the Customer clicks a link to view a Book. The system retrieves the Book details and displays the Book Details screen.

ALTERNATE COURSES:

***Book not found**: The system displays a Book Details Not Found screen.*

- Bookstore home page is much better than bookstore main area.
- The text description now describes the customer/system interactivity much better; it uses active voice.
- The description is less vague and general.

Link Use Cases to Requirements

11. It must be possible for the user to post reviews of their favorite books; the ***review comments*** *should appear on the book details screen. The review should include a* ***customer rating*** *(1–5), which is usually shown along with the book title in* ***book lists****.*

a. ***Book reviews*** *must be moderated—that is, checked and "OK'd" by a member of staff before they're published on the website.*

b. *Longer reviews should be truncated on the book details screen; the* ***customer*** *may click to view the full review on a separate page.*