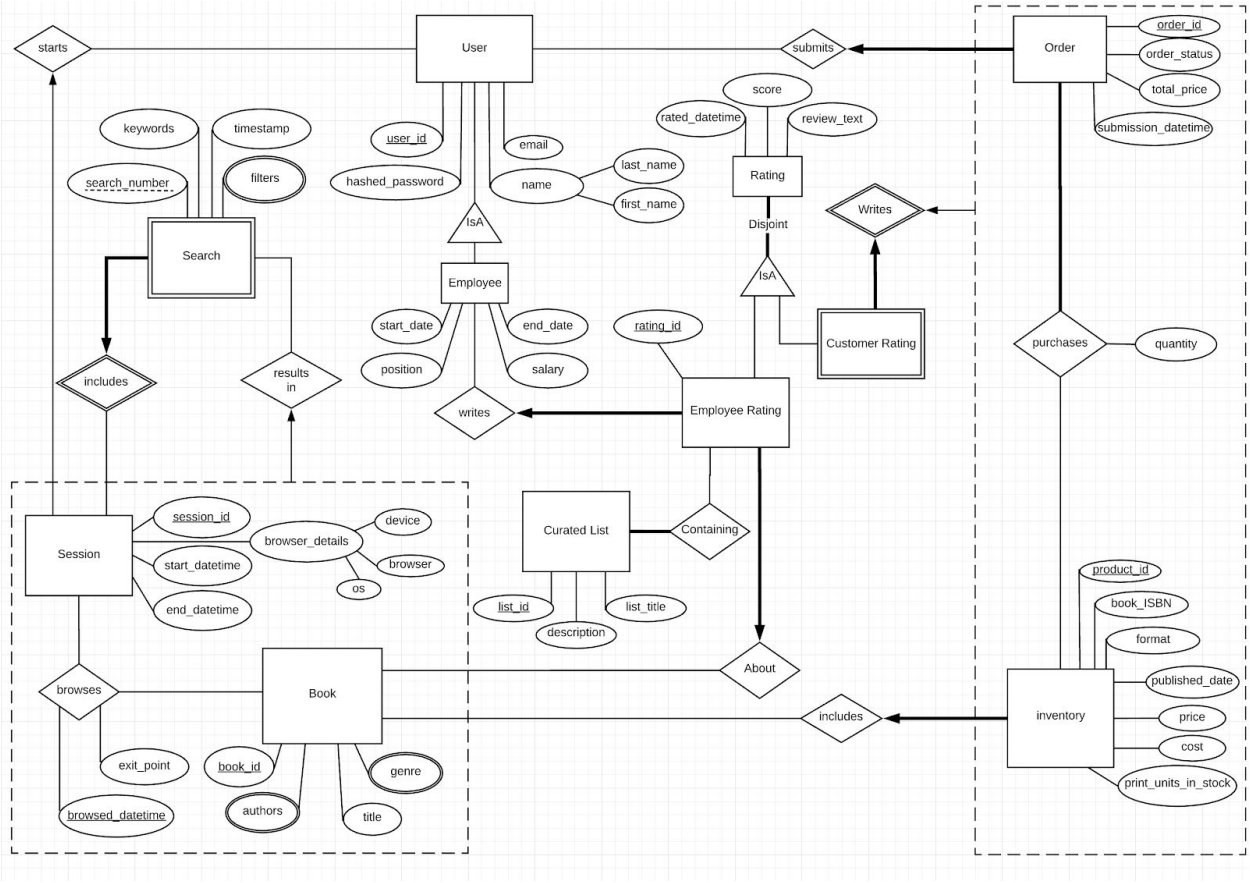
Group members:

Jordan Clark
Alex Elias
Kai Hirota

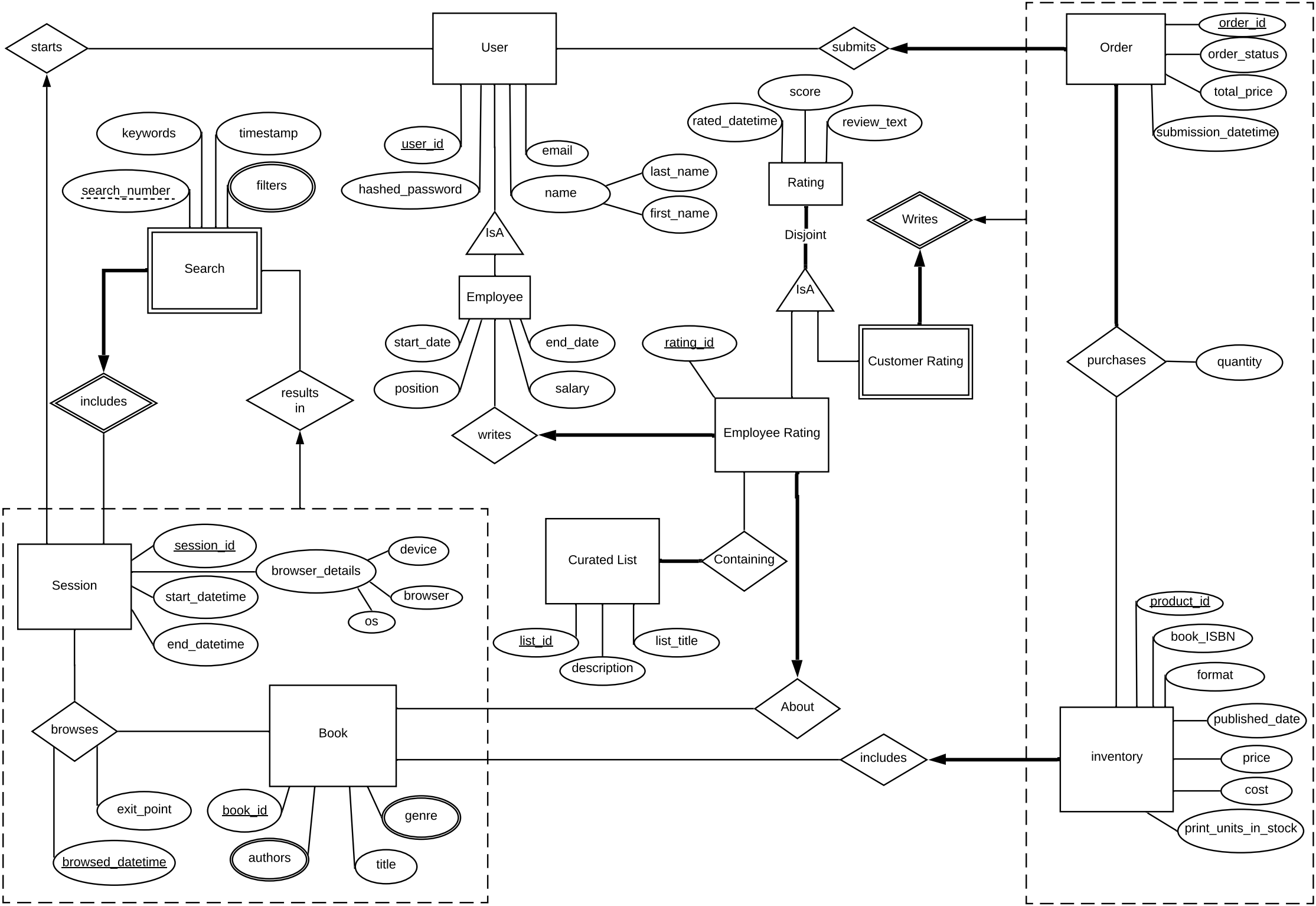
ER Diagram

Link (final version tab at the bottom)



Assumptions

- There are no physical stores, and the organization that is being mapped in the above diagram is an e-commerce website that sells books.
- Employee reviews & curated lists
    - Employees can rate and review books that they have not purchased
    - Curated lists can only be created by employees.
    - Each of the items an employee adds to a curated list must also be reviewed by the employee.
    - A single curated list can be made by several employees.


starts
User
submits
Order
order_id
order_status
total_price
submission_datetime
keywords
timestamp
search_number
filters
user_id
hashed_password
email
name
last_name
first_name
score
rated_datetime
review_text
Rating
Disjoint
IsA
Writes
Search
Employee
start_date
end_date
position
salary
rating_id
Customer Rating
includes
results in
writes
Employee Rating
purchases
quantity
Session
session_id
browser_details
device
browser
os
start_datetime
end_datetime
Curated List
Containing
list_id
list_title
description
product_id
book_ISBN
format
published_date
price
cost
print_units_in_stock
inventory
browses
Book
About
includes
exit_point
book_id
genre
authors
title
browsed_datetime

Entity Definitions

User

Users are people with registered accounts, (that are signed in and so are trackable).

Attribute Name	Constraints	Descriptions
User_id (Primary Key)	Not null, Unique	Unique ID assigned to users that signed up to an account.
Hashed password	Not null	Encrypted password used to log in.
email	Not null, Unique	The user's email address needs to be unique so the same user does not make multiple accounts.
Name (first, last)	Not null	For users to identify with their account easily.

Employee(User)

Inherits all attributes from ***User*** and is identified by the primary key user_id. It is useful to have a separate ***Employee*** entity not only to keep details about their employment but also to allow for only ***Employee***s to create ***Curated Lists***. This would not be possible if they were only ***User***s. It is also necessary to have a separate ***Employee*** entity to prevent users from creating curated lists.

Attribute Name	Constraints	Descriptions
Start Date	Not null	First date of employment.
End Date		Can be null for employees that are currently employed.
Position	Not null	The latest position of the employee.
Salary	Not null	For payment of employee purposes.

Order

This is a container for a purchase of one or more ***Inventory*** items.

Attribute Name	Constraints	Descriptions
Order_ID (Primary Key)	Not null, unique	Unique identifier for each order.
Submission_dat	Not null, Default	Time and date that the order was submitted.

etime	CURRENT_TIMESTAMP	
Order_status	Not null, Default ‘order received’	Could be a custom type - Order received, pending payment, shipped, delivered, returned, etc.
total_price	Not null (w/default the ***Inventory Item*** price for that purchase)	Total price (to be) paid by the customer - could be altered by a coupon, bundle deal or employee discount.

Inventory

The entity ***Inventory*** exists due to the fact that books have different versions and come in a variety of formats. An instance of this entity is a unique instance of a ***book***, with respect to the edition and the format (physical, digital, file type). The main difference between the entities ***books*** and ***inventory*** is that the former does not differentiate versions, while the latter does.

Attribute Name	**Constraints**	**Descriptions**
Product_id (Primary Key)	Not null, unique	Unique ID associated with every version of every book that the business ever sells.
book_ISBN	Unique	International Standard Book Number of the book. This attribute is optional, and is not used as the primary key because privately published books do not have ISBNs. Unique constraint is enforced because books have unique ISBNs for each format of the book.
Format	Not null, custom type constraint (should be one of ‘epub’, ‘hardcover’, ‘softcover’, ‘PDF’...)	Physical book, or the file format of the ebook (PDF, ePUB, MOBI, etc).
print_unit_in_stock		Optional attribute. Indicate units that are in stock and can be purchased. Null for non physical books that there is no limit on licenses.
published date	Not null	Published date of the book.
Price	Not null	Price the inventory item is sold for (to customer)
cost	Not null	Price the inventory item is bought for (from suppliers)

Book

The reason OBS is here. Both authors and genres could have been their own entities, but for the purposes of this assignment custom types work fine. This is an entity that models a book, or a story - one writing. This means the ebook of “Pride and Prejudice” and the hardcover original edition “Pride and Prejudice” are seen as the same book.

Attribute Name	Constraints	Descriptions
Book_id (Primary Key)	Not null, unique	Unique ID assigned to each unique book that the business has ever sold, ignoring format or file type.
authors		Multi-valued attribute for books that have multiple authors.
title	Not null	Title of the book.
genre	Custom type enum (‘sci-fi’, ‘romance’, ‘non-finction’, ‘fiction’, ‘zombie’, …)	Multi-valued attribute because some books fall under multiple genres. For example, *Zero to One,* written by Peter Thiel, can be categorized as non-fiction, entrepreneurship, economics, self-help, and so on.

Session

A ***Session*** refers to a chain of webpages that a visitor navigates through within a website, which begins when the visitor arrives, and lasts until the visitor leaves. This is useful to track what ***User***s look at for improving search results and recommendations. start_datetime is not null as there will always be a start datetime at creation, but end_datetime can be null for ***Session***s that are still active. browser_details are recorded to perhaps gain some more information on customers to better tailor results to their demographic. For example, click-through rate or conversion rate of books can be compared across different device profiles (mobile vs desktop).

Attribute Name	Constraints	Descriptions
Session_id (Primary Key)	Not null, Unique	Unique identifier for a session.
start_datetime	Not null, Default CURRENT_TIMESTAMP	Timestamp for the beginning of the session.
end_datetime		Timestamp for the end of the session.
Browser_details (browser,		Client-side metadata.

OS, device)		

Search

This entity captures all the searches done by users/visitors. ***Search*** is defined as a weak entity of session because a ***Search*** can only exist when a ***User*** searches within a ***Session*** that is in progress, or jumps directly to the search result page via a URL. In the latter case, when a ***User*** jumps directly to the search result page, then a new ***Session*** is created. Keywords can be null as you may just be filtering the entire inventory.
The primary key of ***Search*** will become a composite key: (search_number, session_id).

Attribute Name	Constraints	Descriptions
Search_number (discriminator)	Not null	Discriminator of the search subclass.
timestamp	Not null, Default CURRENT_TIMESTAMP	Timestamp of when the search was submitted.
keywords		The text entered in the search bar.
filters		Filters used (multi valued attribute)

Rating

This is an abstract definition of rating tables (which is why it has *disjoint total participation*). A ***Rating*** has a score, and an optional review (along with a recording of when that rating was made). Since it is abstract (cannot have any rows), there is no need for a primary key.

Attribute Name	Constraints	Descriptions
score	Not null	A (0 - 10) score for the book
Review text		The body of the review
Review datetime	Not null, default CURRENT_TIMESTAMP	Timestamp that the rating was created

Customer Rating (Rating)

This is a weak entity that is dependent on (***Order*** purchases ***Inventory***) - as a customer cannot review a book that they have not purchased an inventory item that refers to that book (that is a format of that book). This was decided because if it is defined by the ***Inventory*** item in an ***Order***, (a single instance of a relationship between ***Order*** and ***Inventory***) then it associates with exactly one user and exactly one inventory item, where the user has bought the ***Inventory*** item (as per OBSs requirements).

Its implied composite key would be (***Inventory***.product_id, ***Order***.order_id).
It inherits score, review_text and rated_datetime from Rating.

Employee Rating (Rating)

From the perspective of users and visitors, (in other words, in the interface) ***Rating***s by ***Employee***s and other ***User***s may appear indistinguishable. However, from the perspective of the business, employee and user-written ratings are treated as separate subclasses of the same parent entity. For example, if the organization wants to compute the average number of ***Rating***s per book, ***Employee Rating***s should not be included since they give no insight into ***User*** activity. This also allows ***Employee***s to ***Review*** books that they have not purchased themselves. (which also means that a primary key is needed, since it can no longer rely on purchase information)

Attribute Name	Constraints	Descriptions
Rating id (Primary key)	Not null, unique	Unique identifier for Employee Rating

Curated List

A curated list is a collection of ***Book***s that employees can create (and can be a collaboration of multiple ***Employee***s). If for instance, an employee wants to curate a list of Harry Potter books, the list should only contain unique books in the series, instead of every version of each book in the series. In other words, *Harry Potter and the Sorcerer's Stone,* the first book in the series, should only appear once, not n-times (where n = number of available versions e.g. PDF ePub softcover hardcover).

Attribute Name	Constraints	Descriptions
List_id (Primary key)	Not null, Unique	Unique identifier for curated list
description		Description of curated list
list_title	Not null	For users to easily identify the curated list

Relationship Definitions

Inventory includes Book

Every instance of ***Inventory*** will be associated with exactly one instance of ***Book***, while each instance of ***Book*** can be associated with 0 (the book exists but is not sold by OBS) or many (*Harry Potter and the Sorcerer's Stone [Hard copy], Harry Potter and the Sorcerer's Stone [Collector's edition], Harry Potter and the Sorcerer's Stone [Electronic edition])*.

Order purchases Inventory

Every ***Order*** includes at least one purchased ***Inventory*** item, but one item can be purchased by (included in) 0 or many ***Order***s.
Quantity is the number of that inventory item that have been ordered.

User submits Order

This is how a ***User*** buys ***Book***s. Every ***Order*** includes exactly one ***User***, but one ***User*** can submit 0 or many ***Order***s (each containing 0 or many ***Inventory*** items - each inventory item referring to exactly one ***Book***).

Session browses Book

A ***Session*** can browse 0 to many ***Book***s, and a ***Book*** can be browsed by 0 to many ***Session***s.
A ***Session*** browsing a ***Book*** means that the user is interacting directly with that ***Book***s webpage, and should be recorded so recommendations can be made to customers with few purchases based on browsing history instead of purchase history. exit_point is where the user went from the ***Book***s page and allows us to see how successful a ***Book***s page is at driving purchases and account creation - useful for improving the website. When this is made into actual tables, this would become a table of its own, with a composite key comprised of (session_id, book_id, browsed_datetime) as a session can only be performing one action at a time.

Search results in (***Session browses Book***)

This relationship models the relationship between searches and visits to product pages.
A ***Search*** can result in a user browsing a book during their session. One search could result in 0 or many ***Book***s being browsed. When a ***Book*** is browsed, its entry point can either be 1 ***Search*** or it could be some external source (so it has a 0 or 1 relationship).
In the tables, ***Session*** browses ***Book***s implied table would have an optional foreign key (***Search***.search_number, ***Search***.session_id), meaning it defines a relationship with ***Search***.
This models the behaviour where a ***Search*** can be related to a “browses” relationship from a ***Session*** it does not belong to (if a search result is linked from one person to another). This may give OBS insight to which of its users are influential and so more valuable.

Session includes Search

Every ***Search*** must have been created by a ***Session***. A ***Session*** can have zero or many ***Search***es (if its entry-point was a book page, and they browse only that page or other pages by clicking recommendations).

Employee Rating about Book

Employee Rating makes reference to ***Book***s instead of ***Inventory*** because soon-to-be-released ***Book***s can also be reviewed by ***Employee***s. In addition, (similar to ***Customer Rating***) an ***Employee Rating*** must be about one and only one ***Book***. (this is

indirectly implied in ***Customer Rating***, but here it is explicit). It also does not make sense that a ***Book*** has multiple ***Rating***s for different formats that it is offered in.

Employee writes Employee Rating

An ***Employee*** can write 0 or many ***Employee ratings*** but each ***Employee rating*** must be written by exactly one ***Employee***. Unlike ***Customer rating***, employees can write ratings on books they have not purchased.

User starts Session

A **Session** is started by a **user**. A ***User*** can create many ***Session***s but a session can be made by at most 1 ***User***. Could be 0, as a person who uses the website while not logged in may create a ***Session*** without an associated ***User*** (linked to a ***User*** when they log in).

Curated List containing Employee Rating

A **curated list** references **Employee ratings.** Every ***Curated List*** includes at least one ***Employee Rating***, but one ***Employee Rating*** can be in 0 to many ***Curated Lists***. Because an ***Employee Rating*** is about exactly one ***Book***, a ***Curated List*** of ***Employee Rating***s defines a list of ***Book***s (where every ***Book*** has an ***Employee Rating***).

(*Order purchases Inventory*) *writes Customer Rating*

A ***Customer Rating*** is written by (***Order*** purchases ***Inventory***) as (***Order*** purchases ***Inventory***) is associated with exactly one ***User***, and exactly one ***Book***. This means A ***User*** can only have a ***Rating*** of a ***Book*** if that ***User*** has purchased an ***Inventory*** item that references that ***Book***. This also allows users to rate each ***Inventory*** item exactly once, after they have purchased it. This is needed for customers and not for staff as customers may have experiences with the specific format rather then the story (perhaps the ebook format was glitchy). This can still appear in the interface as what book was reviewed, and as further information what format the book that was reviewed was in. (for example from personal experience some textbooks work much better on e-book readers then others).
Every rating must then have exactly one (***Order*** purchases ***Inventory***) associated with it and each (***Order*** purchases ***Inventory***) can have up to one ***Customer Rating*** associated with it.