# Project Excersise Book Store

## Database/Java Model

* Article
  + Id
  + Title (length 100, required)
  + Price (0,00, required)
  + Id of Supplier (string, length 100, required)
  + Abstract class, MappedSuperClass
* Book
  + Author (length 100)
  + Isbn (bv 978-90-1234-567-8, validate format, required, unique )
  + Number of pages
* LP
  + Artist (length 100)
  + Genre: classic, pop, rock, dance, r&b, hiphop (enumeration)
* Game
  + Publisher (length 100)
  + Minimum age
  + Genre: MMORPG, RPG, FPS, RTS, RACE (enumeration)
* Order
  + Id
  + Order date (Timestamp)
  + Articles (1 order has multiple articles and one article can be ordered through multiple orders )
  + The current price and the ordered quantity each article should be kept
  + User (FK)
* Review
  + Id
  + Rating (0-10)
  + Description
  + User (FK)
  + Article (FK)
* User
  + Id
  + Firstname
  + Lastname

## Functional

1. **UC1**: Create your database model using sql (save the sql scripts in your project)
2. **UC2**: When visiting the website, you should first be presented a dropdown where you can choose a user (we mock login-functionality)
3. **UC3**: We start with displaying the articles. (Make sure you have a sql script for the initial load of your data, save this script in your project)
   * Create a page with an overview of all the articles, sorted (via database) by type, title and price
   * Create an overview page for each type of article (a page for Books, a page for LP’s, …). Sort by title, author/artist/publisher, price
   * A detail page for each type of article ( every property should be shown )
     + The detail page should have a list with reviews
4. **UC4**: Create an article
   * With both front-end and back-end validation
5. **UC5**: Update an article (try to reuse the functionality of the creation of an article)
   * With both front-end and back-end validation
6. **UC6**: Remove an article
   * This functionality should be implemented on the detail page
7. **UC7**: Search form:
   * You should be able to search on id, type, title AND price (max and min)
   * At least 2 criteria should be applied, if not, show an error message
   * You should be able to sort on 1 of the 4 criteria (selection by dropdown, so sorting you be done by using database queries, no front-end sorting!)
   * This functionality should be implemented by dynamic queries according the applied search parameters
8. **UC8**: Basket
   * On the detail page, you should be able to add an article in your basket. An article can be ordered multiple times, so provide a input field where you can give the number of articles wanted (by default 1 should be filled in)
   * You should be able to order a number of items of a certain article at one time and add some more items of the same article later on
   * Create an overview page with
     + Quantity and price per article
       - column for type
       - column for title
       - column for quantity
       - column for unit price
       - column for total price
     + Number and price per type (book, lp, game)
       - column for type
       - column for quantity
       - column for total price
     + Total price and quantity of articles in the basket
     + A button to place an order
       - Saves the order in the database
       - Empties basket
     + A button to cancel the order
       - Empties basket
9. **UC9**: SupplierService
   * When placing an order, the quantity of the each ordered article should be forwarded to a Mock Supplier Service
   * If an exception is thrown when forwarding an order of a specific article to the supplier. This article should be removed of the order and an error message should be shown. (Mock this by throwing an exception when the quantity is a multiple of 3)
10. **UC10**: A user should be able to mark articles as their favorite. A user can have multiple favorite articles, an article can be the favorite of multiple persons (many-to-many relation in your database). Implement this functionlaity on the detail page. ( try to implement a ‘toggle’ )
    * Watch out! We don’t want a java class for the many-to-many relation. A user should have a list of favorite articles. An article does not have notions of the users that have selected the given article as a favorite.

## Extensions

* **UC11**: Whooops, a title should have a length of 255 characters. (No, don’t drop your database, use an alter table) Don’t forget to update your validation.
* **UC12**: Add a search title where you strip the title of special characters, replace captical cases to lower cases, …
  + When searching on a title use this property. The value searched on, should be given the same treatment (strip special characters, …)
  + The search title should automatically be calculated and saved when inserting or updating an article (prepersist, preupdate)
* **UC13**: Add audit data to all of the entities (review, article, order)
  + Audit (is an embeddable)
    - Timestamp and user of creation
    - Timestamp and user of last update
    - Shoud be saved automatically
* **UC14**: Add Versioning (@Version) on your entities (and learn why this is used)
* **UC15**: Replace your mock supplier service by a rest call and use SoapUI to implement your supplier service
* **UC16**: Add paging (backend) to the overview pages (max 10 articles per page)