Feature: Price Range Sorting

Developer: Parth Thanki
Date submitted: 05/05/2024
Reviewer: Khuong Nguyen
Date Reviewed: 05/04/2024

Major positives

- The feature aligns with its original design goal of Reliability in that it will provide reliable data. In addition, the implemented pagination logic and filtering criteria in the developer design demonstrate a focus on handling large datasets. Filtering results by page number, page size, and price range reduces the server load and response time.

- The developer follows the BRD design and implements the success and failure cases accordingly along with specific logging instructions and business requirements such as the max and min values where the user cannot enter anything less than 0 for the bottom price and anything more than 1,000,000.

- The Sequence Diagrams that show Price Range Sorting Success and Failure follow the requirements inside of the BRD very well, and it shows the developer has clear thoughts on the flow of the feature.

Major negatives

- The values passed through the parameter need more validation to check for length and validity to prevent hackers from inserting malicious inputs that can ruin the database.
    - Checking the valid length of parameters
        - Item Names contain a maximum of 250 characters.
        - The controllers also need to validate that the bottom price is not receiving any bottom price < 0 and the top price does not receive any value > 1,000,000.00

Unmet Requirements

- The developer does not have any test cases that will test the feature when many users are using the feature at the same time since this feature will be available to both authenticated and unauthenticated users.

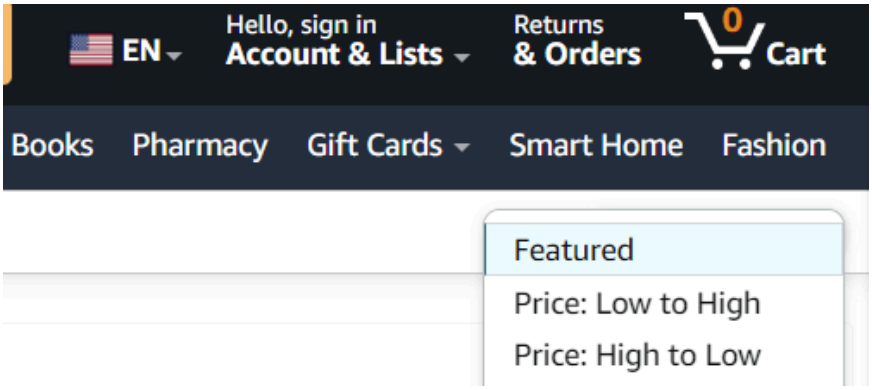
- The code lacks an implementation to ensure or verify that results are displayed within 3 seconds and notify the user of the next action if the request does not return the response within those 3 seconds(Craft Verify BRD: Feature #4 Price Range Sorting: Success Outcome Bullet Point #1).
- The backend service needs to enforce that no more than 100 pages worth of items can be returned at the same time since that will majorly affect feature performance under stress tests(Craft Verify BRD: Feature #4 Price Range Sorting: Success Outcome Bullet Point #4).

- Success Outcomes:
    - An authenticated/ non authenticated user successfully sees the sorting results of craft goods, auction goods, and craft classes in under 3s.
    - Results are within the top and bottom prices.
    - Results are shown in up to 10 items/classes per page format.
    - Results pages are from 0 up to 100.

Design Recommendations

- The feature needs to be designed and implemented with a better UI for the front end that could satisfy both mobile and desktop views.
    - This can be implemented if the developer could utilize frameworks or libraries, but they need to make sure the technologies are approved through the DAR report.

- The price range sorting feature currently only allows the user to view the items from the lowest price to the highest price, so it would benefit greatly if the user is allowed to see other views through different types of sorting such as from the highest to lowest, A - Z (ascending alphabetical order) or Z - A (descending alphabetical order), and adding a different type of sorting will follow the developer design goal of Reusability:
    - EX: Amazon

- 

- In order to make sure the backend project is safe from stack buffer overflow and returning invalid data due to invalid parameters such as bottom price, top, price, page size, and page number, the developer should add validation into service and controller layers to prevent attacks and invalid fetch references from unauthenticated and authenticated users.
    - Add if conditions to check whether the bottom price is more than 0 or the top price is less than 1,000,000.00. Throw the exception and return the appropriate response to notify the user.
    - The developer needs to make sure the page size and page number are valid and it follow the restrictions that they have implemented in the front end: page size == {possible items per page}, and page numbers should not exceed 100 pages.
    - Adding these validations will ensure the developer design goal of Reliability since returning the malicious fetch request as soon as possible will allow real users being able to retrieve the response as fast as possible.

Test Recommendations

- I recommend the developer create some stress tests for the front end to see whether the feature can handle more requests.

    - The developer should create a cypress test case where the Sort Item button is being pressed by an unauthenticated user multiple times at a preferably fast pace, and another test case for authenticated user.

- The developer should create a frontend test to check whether the result of at least one item matches the data inside the database to ensure the data does not get corrupted on its way over to the frontend project.

- The developer also needs to perform these tests for the backend projects because we should not believe in our users
    - Test the boundaries of your price range filters, such as prices at exactly the minimum value for the bottom price and maximum value for the top price to ensure they are returning the appropriate results.
    - Test the feature when it comes to invalid inputs, such as negative page numbers, overly large page sizes, and invalid price ranges, and the bottom price is greater than the top price.
    - The test returns an empty result without triggering any error or exceptions set up by the developer.