

Product Review

System

Final Project

Emma Richardson

Emma Richardson

## Product Review Sorting and Filtering System Problem

statement:

The current system used for customers to view products and their respective reviews does not allow for organizing such products by their review rating or to see common topics brought about in customer reviews. Buyers may wish to sort or filter inventory results by rating. They may also seek to understand reviews of different aspects of a given product.

Functionality:

The objective of the proposed system is to give customers the tools to sort and filter products based on the experiences of buyers, thus improving their shopping experience. In addition, the proposed solution will offer the ability to view different categories commonly mentioned in reviews.

To achieve such a system, the following requirements must be met:

- User review submission – The system must allow for users to submit their review by selecting a rating out of five, along with the option to add context via plain text to enhance or further explain their conclusions. Submissions should also include categories for users to select based on their experience, such as “Good Quality”, “Would Buy Again”, or “Would Not Recommend”
- Filtering product results – The system will give users the options of filtering their search by the above-mentioned review categories. The results of the chosen category should be a relatively close match to the search, meaning most customer reviews include said category.
- Sort products based on ratings – The system must be able to sort products based on the rating out of five and display them from highest to lowest.

Target Customers:

Users of this system will be online shoppers, who often filter and sort products for which they are shopping based on the reviews of buyers of the products.

#### Technical Requirements:

##### Software:

- This system will require a database to store customer reviews and correlate reviews to products.
- The proposed system will also require a programming language to produce and sort results based on customer selections.
- The network requirements include a web server host, such as AWS.

#### Product Review Sorting and Filtering System Development

##### Approach:

The development of the proposed system will utilize the programming language Python and database using SQL, utilizing multiple tables to store relevant data.

##### Development Plan:

The schedule for development and implementation can be broken down into steps. These steps are listed below:

1. Strategic Planning and Analysis
  - Define the structure of the system as directed by analysis of goals, budget, and abilities
  - Estimated timeline: 2 weeks
2. System Development
  - Develop database and backend programming
  - Estimated timeline: 3 weeks
3. Frontend/User End Development
  - Develop the user end of the system and implement sorting and filtering -  
Estimated timeline: 3 weeks
4. System Testing
  - Test system for any potential errors and/or pain points
  - Estimated timeline: 2 weeks
5. System Implementation and Deployment
  - Deploy on server host for customer use
  - Estimated timeline: 2 weeks

#### Problem Statement

Searching through items when shopping online is often time-consuming when certain features are unavailable. Features that simplify the process are sorting and filtering based on selections made. If customers only wish to browse through items that have a higher rating, offering a filter with that selection available makes the shopping process more enjoyable and efficient. Additionally, when viewing a product and its reviews from past customers, filtering the reviews by common words allows for a better understanding of other customers' experiences.

#### Glossary of Terms

AWS (Amazon Web Services)	Cloud-based website host platform
Database	System used to store, organize, and access data
Python	Programming language
SQL	Programming language used in combination with databases
Uptime	Describes time a system is available

#### Functional Requirements

No.	Priority Weight	Description
REQ-1	High	Filtering product results: Users should be able to view items based on filter selections
REQ-2	High	User review submission: Users should be able to submit their reviews to the site and select categories that describe their experience
REQ-3	High	Sorting products based on ratings: Users should be able to sort items based on reviews
REQ-4	Medium	Categories of reviews: Users should be able to choose a category to see reviews that fit into that category

Nonfunctional Requirements

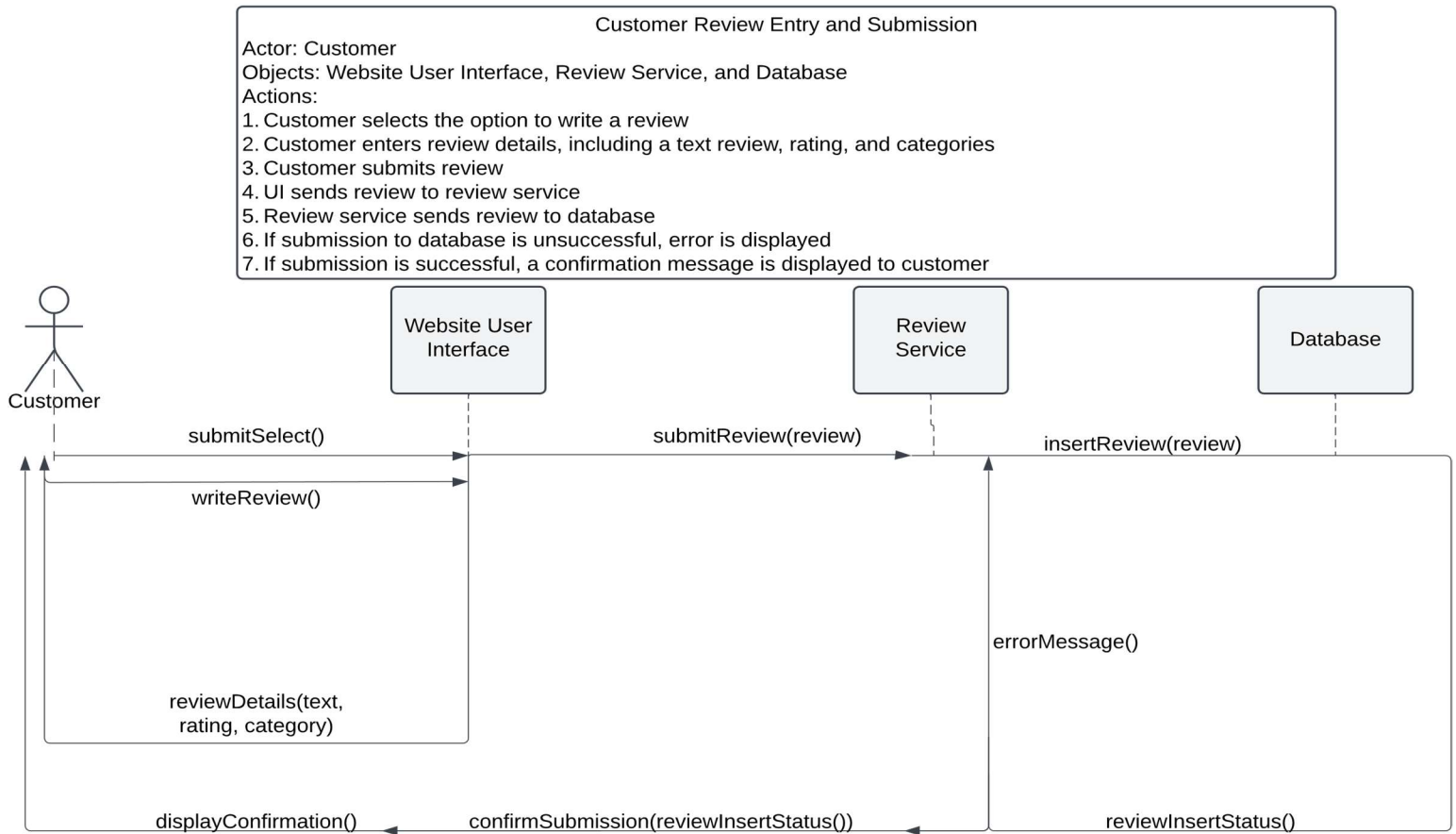
No.	Priority Weight	Description
<b>Functionality</b>		
REQ-1	High	Sorting and filtering results are accurate
<b>Usability</b>		
REQ-2	High	Site is simple to use, demonstrating an intuitive layout
<b>Reliability</b>		
REQ-3	High	The site should have a high uptime to ensure customers are able to use the site and its features.
<b>Performance</b>		
REQ-4	High	The site should have a quick response time to selections made by customers
<b>Supportability</b>		
REQ-5	Medium	The site should be simple and easy to update or otherwise modify

Technical Requirements

No.	Priority Weight	Description
REQ-1	High	SQL database for storage of reviews and ratings
REQ-2	High	Backend developed with Python for review submission, sorting and filtering
REQ-3	Medium	User-friendly interface that allows customers to navigate the site and its items
REQ-4	Medium	AWS used to host site and allow for customer use

## System Sequence Diagram

Emma Richardson



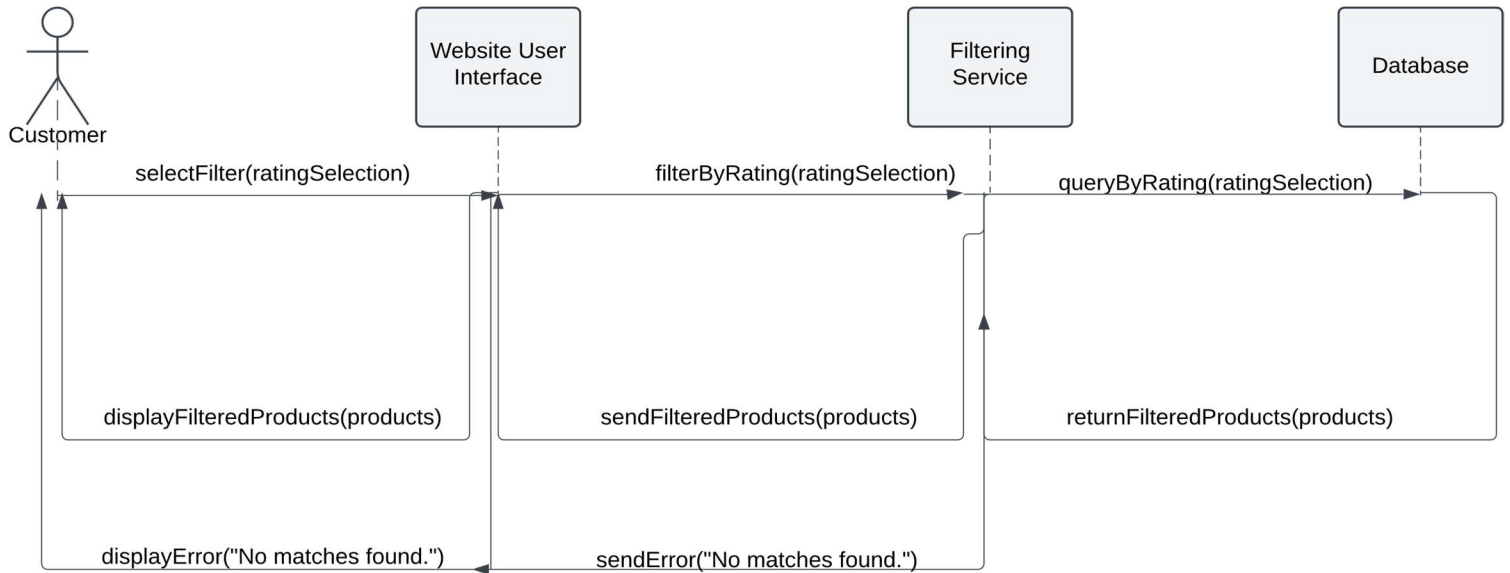
### Filtering Products by Rating

Actor: Customer

Objects: Website User Interface, Filter Service, and Database

Actions:

1. Customer selects the rating by which they'd like to filter
2. UI sends rating selection to filtering service
3. Filtering services send rating selection to database to query the ratings within
4. If database does not contain any matches, an error message is displayed to customer on UI
5. If there is a match, database returns products matching query to filtering service
6. Filtered products are sent to UI
7. Filtered products are displayed to customer



# User Interface Specifications

## Customer writes a review



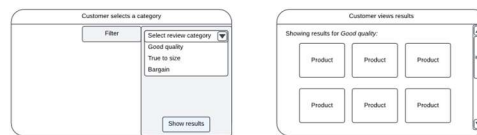
### User effort estimation:

Item page, rating selection, category selection, text entry, confirmation page

Clicks: 5-8

Keystrokes: 0 - 250

## Customer filters by review category



### User effort estimation:

Product page, filter bar, results

Clicks: 3

Keystrokes: 0

## Customer sorts products by rating



### User effort estimation:

Product page, sort bar, results

Clicks: 3

Keystrokes: 0



Customer filters products by rating

Customer selects to filter by rating

Filter Select

Rating  
Size  
Price

Customer selects a rating out of five

Filter Select rating

★★★★★  
★★★★  
★★★  
★★  
★  
Show results

Customer views results

Showing results for Rating

Product Product Product

Product Product Product

User effort estimation:

Product page, filter bar, rating selection bar, results

Clicks: 4

Keystrokes: 0

Customer selects categories for review

Item page

Review Product

Customer rates item

Select rating out of five:

★★★★★

Back Next

Customer selects applicable review categories (optional)

Please select all that apply:

☒ Good quality  
☐ True to size  
☐ Bargain

Back Next

Customer enters text review

Categories selected

Good quality

Enter your review (optional)

Back Submit Discard

Confirmation screen appears

Review submitted

Thanks for your feedback!

Home

User effort estimation:

Item page, rating selection, category selection, text entry, confirmation page

Clicks: 6-8

Keystrokes: 0 - 250

# Traceability Matrix – Emma Richardson

No.	Priority Weight	Description
REQ-1	High: 5	User review submission: Allow users to submit their rating, text, and categories
REQ-2	High: 4	Filter products based on ratings: Allow users to filter product results based on a selected rating from submitted reviews
REQ-3	High: 4	Sort products based on ratings: Allow users to sort product results from high to low ratings from submitted reviews
REQ-4	Medium: 3	Filter products based on review categories: Allow users to filter product results based on their selection of a category frequently (2+ times) included in submitted reviews
REQ-5	High: 5	Establish a database that stores user review submissions
REQ-6	High: 5	Implement python to filter and sort data from database and send to user interface
REQ-7	High: 5	Web hosting server for user interface

No.	Description
UC1	Submit review, ensuring rating and category selection is completed correctly
UC2	Test filtering feature, checking that only products with the selected rating appear
UC3	Test sorting feature, checking that the rating of each product is higher than or equal to the one appearing after
UC4	Test filtering by category feature, ensuring only products that have at least 2 reviews that include the selected category appear
UC5	Test database to verify user reviews have been stored and linked to the reviewed product
UC6	Test code for sorting and filtering capability and correctness
UC7	Test deployment on web hosting site to ensure accessibility and function

No.	PW	UC1	UC2	UC3	UC4	UC5	UC6	UC7
REQ-1	5	X						
REQ-2	4		X					
REQ-3	4			X				
REQ-4	3				X			
REQ-5	5					X		
REQ-6	5						X	
REQ-7	5							X

## Product Review System

This system aims to store product reviews and allow users to sort and filter them.

I utilized SQLite for a database, which stores products, users/reviewers, reviews, and categories.

The database structure consists of 5 tables, which are demonstrated below:

Table name	Purpose
products	Holds product details
users	Holds user/reviewer information
reviews	Stores product reviews
review_categories	Stores review tags
review_category_map	Joins reviews and categories

Instructions:

1. Navigate to codespace within GitHub
2. Go to this link: <https://emmarichardson6148.github.io/Product-Review-System/>
3. Now you can enter your reviews.

# Product Review System

Emma Richardson

## Slide 1

This slide demonstrates the setup of the database that stores and correlates customer reviews and product information.

- @EmmaRichardson6148 → /workspaces/Product-Review-Database (main) \$ sqlite3 database.db  
SQLite version 3.45.3 2024-04-15 13:34:05  
Enter ".help" for usage hints.  
sqlite> .tables  
products                  review\_category\_map  users  
review\_categories      reviews  
sqlite> SELECT \* FROM products;  
1|Daytime Blouse|A casual blouse for daily wear  
2|Elegant Dress|A stylish evening dress  
3|Comfortable Hoodie|A cozy hoodie for lounging  
sqlite> .exit
- @EmmaRichardson6148 → /workspaces/Product-Review-Database (main) \$

## Slide 2

The next image shows the process of adding reviews to the database.

• @EmmaRichardson6148 → /workspaces/Product-Review-Database (main) \$ python add\_reviews.py

Reviews submitted.

• @EmmaRichardson6148 → /workspaces/Product-Review-Database (main) \$ sqlite3 database.db

SQLite version 3.45.3 2024-04-15 13:34:05

Enter ".help" for usage hints.

sqlite> SELECT \* FROM reviews;

1|1|1|5|Great blouse!|2025-03-20 01:25:56

2|2|1|4|Pretty good, but could be better.|2025-03-20 01:25:56

3|3|2|3|Average quality, expected more.|2025-03-20 01:25:56

4|1|1|5|Great blouse!|2025-03-20 01:30:57

5|2|1|4|Pretty good, but could be better.|2025-03-20 01:30:57

6|3|2|3|Average quality, expected more.|2025-03-20 01:30:57

7|1|1|5|Great blouse!|2025-03-20 01:36:09

8|2|1|4|Pretty good, but could be better.|2025-03-20 01:36:09

9|3|2|3|Average quality, expected more.|2025-03-20 01:36:09

sqlite> █



### Slide 3

This last slide demonstrates the program's ability to sort and filter the reviews from the database from highest to lowest on one product.

● @EmmaRichardson6148 → /workspaces/Product-Review-Database (main) \$ python filter\_products.py

Filtered & Sorted Products by Rating:

Product ID: 1, Name: Daytime Blouse  
Description: A casual blouse for daily wear  
Average Rating: 5.00

Product ID: 2, Name: Elegant Dress  
Description: A stylish evening dress  
Average Rating: 4.00

Product ID: 3, Name: Comfortable Hoodie  
Description: A cozy hoodie for lounging  
Average Rating: 3.00

● @EmmaRichardson6148 → /workspaces/Product-Review-Database (main) \$ █