Project Report

fa24-cs411-team081

1. Please list out changes in the directions of your project if the final project is different from your original proposal (based on your stage 1 proposal submission).

In our final project, we made some adjustments to the original proposal by removing features that were challenging to implement due to the lack of appropriate data sources. Specifically, we decided to eliminate the browsing history feature, which was initially intended to allow users to revisit and manage previously viewed items. This feature required a dataset to track user activity and store browsing records, but we could not find a suitable or realistic database to support this functionality within the scope of our project. As a result, we chose to prioritize features that could be more effectively implemented with the available resources.

Additionally, we removed the real-time trending products feature, which was originally designed to highlight weekly trends based on user interactions, reviews, and sales data. This feature required real-time access to external data sources or complex algorithms for trend analysis, which were beyond the scope of our current resources and dataset availability. While this feature would have enhanced user engagement, we decided to focus on delivering a functional platform with the other planned features that align with our project's objectives.

Despite these changes, we successfully implemented all other core features from the original proposal. These include user profiles with secure authentication, advanced search and filtering options, personalized product recommendations, wish lists, comments and reviews, video integration from YouTube, and the innovative bundle creation feature. By prioritizing these functionalities, we were able to create a robust platform that provides a comprehensive and engaging experience for users while addressing the challenges of choice overload in the beauty and cosmetics industry.

2. Discuss what you think your application achieved or failed to achieve regarding its usefulness.

Our application successfully addressed the primary challenge of simplifying product discovery in the beauty and cosmetics industry. By implementing features like personalized product recommendations, advanced search and filtering, and curated bundles, we provided users with a streamlined, engaging, and efficient shopping experience. Additionally, the integration of YouTube video previews enhanced the decision-making process by giving users visual demonstrations of products. The wishlist and user profile functionalities added a layer of personalization, making the application more practical for users.

However, the application fell short in implementing browsing history and real-time trending products, as we could not find or create suitable datasets to support these features. These omissions limited the scope of user engagement and real-time feedback that we had initially envisioned. Additionally, while our recommendation system works well with the data available, it could be further improved with larger, real-world datasets and more advanced algorithms to enhance accuracy and relevance.

Despite these limitations, our application demonstrated its potential to address choice overload in the beauty market. The features we implemented are both practical and impactful. Moving forward, we would focus on obtaining more comprehensive datasets and improving the backend to support real-time features. This would make the application more robust and further enhance its usefulness for end users.

3. Discuss if you change the schema or source of the data for your application

We made adjustments to our data schema to better align with the features we implemented. For instance, we modified the schema to include separate tables for user profiles, wish lists, product reviews, and bundles. These changes allowed us to organize the data more effectively and enable seamless integration of features such as personalized recommendations, comment tracking, and bundle creation. Additionally, we added fields to support the integration of YouTube video URLs into product details.

While we initially planned to use datasets from Kaggle and the YouTube Data API, we found that the Kaggle dataset did not cover all attributes required for our application, such as detailed user reviews and specific product interactions. To address this, we supplemented it with synthetic data for testing and development purposes. The YouTube Data API integration remained consistent with our original plan, but we focused on storing fetched video URLs in the database for reuse.

4. Discuss what you change to your ER diagram and/or your table implementations. What are some differences between the original design and the final design? Why? What do you think is a more suitable design?

In the original design, we had a single Bundle table that might have included all the necessary data for both the bundle (as a collection) and its associated products. This approach had limitations, and splitting the table into two separate entities (Bundles and BundleProducts) addresses these challenges effectively. By splitting the data into two tables, the system can handle more complex queries and relationships (e.g., fetching all products in a bundle or all bundles containing a specific product) more efficiently. In addition, splitting the table makes it easier to extend functionality for future development.

5. Discuss what functionalities you added or removed. Why?

We decided to remove the following two functionalities from the original proposal due to time constraints, technical challenges, and shifting priorities:

Browsing History

Initially, we planned to implement a browsing history feature, allowing users to track and manage their previously viewed items. However, we encountered technical difficulties related to data storage and user privacy concerns. Additionally, we did not collect the necessary data to support this feature, and as a result, we were unable to create the corresponding tables for tracking user interactions.

Real-Time Trending Products

This is aimed to highlight popular items based on user interactions, sales data, and reviews. Although this feature had potential, it required the integration of complex algorithms and real-time data processing, which would have added significant development time. After further consideration, we realized that the platform could still provide valuable recommendations through a more static set of curated lists based on product categories and ratings.

6. Explain how you think your advanced database programs complement your application.

The advanced database programs used in my application complement the product and user interaction functionalities by ensuring efficient data storage, retrieval, and management. Below is an explanation of how they support the overall system:

Efficient Data Retrieval and Filtering

The `GET /api/products` endpoint allows users to search for products using various filtering options such as search terms, price range, brand, product type, and rating. This functionality is powered by optimized database queries that use indexing and filtering mechanisms to ensure fast data retrieval, even when dealing with large datasets. For instance, products can be filtered by different attributes (price, rating, etc.) with minimal performance degradation, providing a seamless user experience.

Sorting Capabilities: The ability to sort products based on user preferences (e.g., by rating, price, or recency) is implemented using SQL `ORDER BY` clauses, which allow us to sort product listings dynamically while maintaining optimal performance.

Fetching Detailed Product Information

The `GET /api/products/:productId` endpoint retrieves detailed product information, including associated video links and user comments. This requires joining the `products` table with the `comments` table, which is handled using efficient database joins. The advanced database programs ensure that this join operation is performed quickly, even when dealing with a high volume of comments for each product.

One-to-Many Relationship: The `products` table is linked to the `comments` table via a one-to-many relationship, where each product can have multiple comments. By using LEFT JOIN queries, we can retrieve all products, even if they have no comments, and efficiently load the associated data.

Video Links Integration: Video links (from YouTube or a database) are fetched alongside the product details, ensuring that the full product experience is delivered in a single query. Storing these links in a separate table, related to the product via the `productId`, ensures that they can be easily updated or added without altering the core product data.

Comment Management

The `POST /api/product/:productId/comments` endpoint allows users to submit comments, which are then stored in the `comments` table. This data is linked back to the respective product using the `productId`, ensuring proper data integrity. The advanced database systems ensure data validation by checking required fields, such as `userId`, `rating`, and `content`, at the application layer before the data is inserted into the database. This process prevents incomplete or malformed data from being stored, ensuring data integrity and consistency across the platform.

7. Each team member should describe one technical challenge that the team encountered. This should be sufficiently detailed such that another future team could use this as helpful advice if they were to start a similar project or where to maintain your project.

Ke: One technical challenge our team encountered was maintaining consistency in variable naming conventions. Inconsistent names led to confusion and made it difficult to trace errors. For example, using different names for similar data structures or variables in different parts of the code resulted in bugs that were hard to debug, wasting time during extended sessions. From this experience, I learned the importance of clear, consistent naming conventions across the codebase. It not only prevents confusion but also improves readability and maintainability. I also became better at troubleshooting errors by analyzing error messages and tracing them to inconsistent variable names.

Yifei: Using MySQL transactions in Express.js requires careful handling of asynchronous operations. We initially struggled with using callbacks for beginTransaction, commit, and rollback, which made the code difficult to read and error-prone. We then decided to switch to Promises and wrapping these functions allowed us to use async/await, making the code cleaner and easier to debug. For future projects, if they require complex database operations, we think we should plan how to handle transactions early in the design phase.

Tong: One technical challenge I encountered was dealing with mismatched data types between two files. Specifically, one file contained a variable with a tuple of three elements `(x, x, x)`, while the corresponding variable in the other file had a tuple of two elements `(x, x)`, which caused compatibility issues during data merging. To resolve this, I transformed the data by either adding default values to the shorter tuple or removing unnecessary elements from the longer tuple, ensuring both datasets had matching structures. I also implemented data validation checks to ensure consistency. This experience highlighted the importance of standardizing data formats early, automating data cleaning processes, and documenting expected structures to avoid similar challenges in future projects.

Yiyu: One major challenge I faced was handling the YouTube Data API's strict quota limits while integrating video previews into our platform. Initially, every time a user viewed a product detail page without a cached video, we made a new API call to fetch a related YouTube video. This approach worked well during the early stages, but as our user base grew, we quickly hit the daily quota limits. This led to failed video fetches, resulting in missing previews and a poorer user experience. To address this, we implemented a caching system where once a video link was retrieved for a product, it was stored in our database. This meant that subsequent views of the same product didn't require additional API calls, significantly conserving our quota.

8. Are there other things that changed comparing the final application with the original proposal?

In addition to removing the Browsing History and Real-Time Trending Products functionalities, several changes were made to the final application compared to the original proposal due to evolving project priorities and feasibility:

Enhanced Focus on Bundles

In the original proposal, bundles were a secondary feature, but they were emphasized in the final application as a core offering. We realized that bundles add significant value by allowing users to explore complementary product sets, making them a focal point of user engagement.

Improved Video Integration

The original proposal mentioned integrating YouTube video previews. This feature was refined in the final application to ensure seamless video scraping and embedding, enhancing the user experience without adding complexity.

Adjustments in the UI/UX Design

The final application incorporated a more streamlined and minimalist design compared to the initial proposal. This was done to improve usability and reduce development overhead for advanced animations or UI components.

By focusing on feasible and impactful features, the final application delivers a robust and tailored user experience while staying aligned with the core goals of the project.

9. Describe future work that you think, other than the interface, that the application can improve on

Real-time Product Data Fetching and Updates

Currently, the product information on our platform is based on data that has been pre-pulled or statically stored. While this approach provides stability, it limits the ability to reflect real-time changes such as new product launches, stock availability, and pricing updates. To improve user experience and ensure the platform stays up-to-date, implementing a real-time product data fetching and updating feature would be beneficial. This feature would allow the application to automatically fetch the latest product data from external sources like e-commerce sites, product feeds, or APIs. By doing so, the platform would reflect the most current information on product availability, prices, and new releases. It could involve setting up scheduled jobs or using webhooks to instantly update the data whenever changes occur. This approach would help maintain the platform's accuracy and relevance, offering users a seamless and up-to-date experience.

Incorporating a Forum or Community Platform

Adding a forum or discussion platform where users can exchange product recommendations, experiences, and tips would increase user engagement. It could serve as a valuable space for users to share feedback and insights, as well as build a sense of community around the product offerings.

10. Describe the final division of labor and how well you managed teamwork.

The division of labor for our project was clearly defined and well-balanced among team members, enabling us to work effectively and meet our project goals.

Tong Wu implemented the user sign-in, sign-out, and registration functionality, ensuring secure authentication with password encryption and enabling account creation, login, and management. Tong also developed the wishlist feature, allowing users to save, organize, and delete items directly from their user profile. The wishlist integrates with the user profile page, displaying all saved items in one place for easy management.

Yiyu Weng led the development of the search and filtering features, enabling users to locate products quickly and efficiently based on keywords and attributes such as price, brand, and type. Yiyu also implemented the product ranking feature, which allows users to sort and compare items based on criteria like popularity, effectiveness, or price. In addition, Yiyu integrated video previews into the platform by fetching related YouTube videos. This feature enhances the user experience by providing engaging and informative visual content, such as product tutorials or reviews, directly on the product pages.

Yifei Mao developed the product recommendation system, delivering personalized suggestions based on user preferences, budgets, and interactions. Yifei also created the bundle creation and interaction feature, enabling users to group complementary products into curated bundles. These bundles can be shared within the community, fostering collaboration and discovery. Users can interact with bundles by commenting, cloning, editing, or rating them, adding a dynamic and social element to the platform while helping users explore curated product combinations effortlessly.

Ke Jiang contributed by implementing the comments and reviews section, enabling users to engage with the platform in multiple ways. Users can add, delete, and update comments, provide detailed feedback, and interact with other shoppers. Reviews are accessible in two locations: they can be viewed collectively on the product details page, where all reviews for a product are displayed, and individually in the user's profile, allowing users to keep track of their own feedback. Additionally, any comment added to a product is immediately reflected on the product details page and also appears under the user's profile.

Our teamwork was managed smoothly through regular communication, task delegation, and collaboration on shared components, such as backend integration. Each member was committed to their responsibilities, ensuring that tasks were completed on time while maintaining a high standard of quality. This clear division of labor and effective coordination allowed us to deliver a cohesive and functional platform.