Stage 4: Final Report

Bennett Wu [bwu40]

Jeremy Lee [jeremyl6]

Paul Jeong [paulj3]

Shreya Singh [shreya15]

Project Changes

Throughout the project's lifetime, the design was adjusted according to the project's demands and deadlines. While core functionality remained consistent, the contents of each page of the project's website were adjusted accordingly - the direction of the project itself remained consistent to the core issue being addressed, with no significant changes in direction. The changes to the project are listed below for reference.

Original Proposal	**Revised Design**
Search by Category, Search by Item (and associated pages)	Changed to Search by Keyword, Budget, included Product Statistics option, adjusted associated pages
Frequently Searched	Changed to Search History
Entity FoodGroups Primary Key: product_id	Changed to Foreign Key, added additional Primary Key

Overall, few changes were made to the direction of the project, with functionality being adjusted based on project requirements as the project progressed through its lifetime.

Functionality

Achievements and Shortcomings

Our application had the main functionality implemented well. As mentioned in the previous section we managed to implement the login page, sign-up page, profile page, and search pages. We also managed to implement the navigation bar. We believe that by adding capabilities like search by statistics and search by budget, we were able to make our project more valuable than we had originally planned. Thanks to these capabilities, consumers could search for products in a way that helped them identify stores and products that match their budget when shopping. Additionally, we developed a history toggle that gave users the option to decide whether or not to save their search history. We did not implement the page that allowed users to search by

category. Overall, our application provided a way for users to search for grocery items from different stores, compare prices, and other statistics about the product.

We managed to add some features that were more useful than our proposed project such as search by statistics and search by budget. These features allowed the user to search for products in a way that made it possible for users to shop with a budget and find products and stores that fit that budget. We also created a history toggle which allowed the users to control whether they wanted to save their search history. We did not implement the page which was designed to search by category. We also did not allow users to search by store because product statistics allowed users to compare prices from different stores and find the cheapest one.

We did change the database schema from the original proposal. The user database in the original proposal did not have a save history toggle. This is something that we added to the user database so that we can track whether the user wants to save their search history or not. In addition to that, we also added an additional primary key to the FoodGroups table and used the product_id as a foreign key. We did not make any other significant changes to our initial proposal.

Schema and Diagrams

We didn't change the source of our data, but we did need to remove and rename columns so that the data (in the form of a .csv) matched our database schema.

We added a "save_history" flag to the Accounts relation/table and dropped the "store_type" field from the Stores relation/table. We also added a primary key to the FoodGroup relation/table so that it would be its own standalone entity. The "story_type" field in Stores did not end up being used, so we removed it. The "save_history" flag was added to satisfy the triggers project requirement. We wanted to add a trigger for insertions into the SearchHistory table that would check if a user wanted their history to be saved or not so we needed a flag for each user that the trigger could check for. Overall, we didn't change much from the original design. However, the changes we did make were to better satisfy the project requirements, so in that way, we would say the final design was more suitable in that way.

Database

We think that the advanced database programs complemented our application by providing the user with various statistics about the data and ways to filter that data that they could use to better understand the groceries stores they are shopping at. The two advanced database programs were used for the "Search with Budget" and "Product Statistics" features.

The "Product Statistics" feature used an advanced query to find the average price and standard deviation of products in each store, then for each store, iterated through several histogram buckets and counted the number of items in that bucket. We then finally used an advanced query to join those results with the average price and count of the product they were searching for. This gives the user a way to compare the item they are looking for with how it stacks up against the rest of the products in that store. It's also a way to see how prices at each store are distributed.

The "Search with Budget" feature used an advanced query to find the average price of products in each store, then used the number of products they were searching for and their budget to select the best store for their budget. Then found the cheapest item for each item they searched for using another advanced query. This was to help the user filter using their budget rather than just getting all of the items that matched their search as would be in "Keyword Search".

Technical Challenges

One technical challenge that we encountered was that certain git installs automatically convert line endings in Windows. For example, if you commit a file on a Linux or Mac computer in LF format, it will be stored as such in the git repository, but when you apply the changes on some Windows computers it will convert them to CRLF then back when you commit. This normally doesn't cause an issue, but we had a Docker container running our development MySQL database that read .sql files from our repo to initialize the database with our tables, stored procedures, triggers, and data. This container ran Linux, so if the files had CRLF line endings, it would fail to parse the SQL commands and fail to start. We tried many things, and ultimately running: git config --global core.autocrlf false*, to disable this auto conversion seemed to be the fix that resolved the issue.*

Another technical challenge/concern we faced was syncing our save history switch with the save history boolean in our Accounts field. There was no real issue with having our api take no parameters and just negate the value whenever the save history

switch was toggled. Potentially, if there were multiple sessions with the same account and toggling occurred simultaneously it could have been a problem. To sync, we had our api take a save_history boolean as a parameter from the frontend switch value. This way, whatever the frontend displays it sends to the backend to set to the same boolean value.

For some members, learning to work with and implement APIs, as well as create and integrate frontend and backend elements was a technical challenge, as they had no previous experience with application development. Understanding the purpose of APIs and using them to link frontend and backend elements proved to be a technical challenge that created a minor learning curve for these members, which was ultimately overcome either through the usage of online resources and further deliberation with group members that had more experience. Additionally, standardizing the project's API with HTTP codes proved to be another minor challenge, as additional research to understand these codes was required to properly implement them within the project.

Another technical challenge that we encountered was creating the stored procedure for the Search by Budget feature. Creating a stored procedure that was able to read through a list of items proved to be difficult because it involved creating a cursor that was able to do that. In addition to that, we also had to ensure that it had multiple advanced queries and creating the control structure was also slightly challenging. We were able to resolve these challenges by looking through solutions online, reading guides, and brainstorming. To ensure that the feature worked correctly, we first added the query that could find the average prices of all the stores stored in the database, and only looked through stores that were below the mentioned budget. Then we implemented the second part of the query that only searched for items in the stores that were below the mentioned budget that we found using the first query.

Future Work and Potential Improvements

Our application has a functional base that covers the essential objective of comparing grocery prices among different stores. However, an addition that is almost necessary for the project to be applicable to consumers are live prices. There is no reasonable way that we know of how to accomplish this since these datasets are not publicly available, which is where the next step should be.

Another area of improvement is applying store location. Reasonably, consumers aren't going to want to go to a store with attractive prices that are not within a commutable distance. Again, this would require more datasets that aren't publicly

available. A geolocation api would also have to be implemented to only search within a certain radius of stores.

Division of Labor

Labor was distributed throughout the project evenly where possible, but kept related aspects of the project assigned to the same individuals to prevent the need to relearn aspects of the project itself. Additionally, each team member was assigned components of the project based on interest, past experience, and current workload, in an attempt to evenly distribute the work - each team member also contributed to the written and design aspects of the project at each stage. Overall, the team worked well together, communicating with each other when appropriate and setting forward guidelines and expectations for aspects of the project. Each member's contributions are listed in the table below.

Member	**Contributions**
Bennett	- Sourced datasets - Created DDL for database and set up in GCP - Dockerized environment for development, created baseline application boilerplate - Outlined APIs - Created Show Save History functionality - Created Delete User functionality - Implemented basic aspects of Login page
Paul	- Scraped datasets to fit database tables - Inserted scraped rows into database - Implemented Profile Page and corresponding APIs
Shreya	- Created Low-Fidelity UI Mockup - Created Advanced Queries and performed corresponding analysis - Implemented Keyword Search and corresponding elements - Implemented Budget Search and corresponding elements - Implemented Product Statistics and corresponding elements
Jeremy	- Created ER Diagram - Normalized database - Added indices to database where appropriate

	- Created Advanced Queries and performed corresponding analysis - Implemented aspects of Login page - Implemented Sign Up page