CS 411 Final Project Write Up

Link to Demo:
https://www.canva.com/design/DAGl_LfJqjQ/klSXQEgedakoda1GdMayMQ/watch?utm_content=DAGl_LfJqjQ&utm_campaign=designshare&utm_medium=link2&utm_source=uniquelinks&utlId=haf34b1e685

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).

The general direction of our project remained the same in that we created a website that allows users to see the environmental impact of their food choices. In our proposal, we had the idea to allow the user to choose a product and an importer, and to calculate the environmental cost information based on that. In our actual implementation, we have the user input a product and its source location, and calculate the environmental cost based on that. This is slightly different because we could not find the importer data that we would need to implement the first approach. We also decided to simplify our graphic visualization of the product traveling from the source location to the user's location. Instead of showing different sources (importers) and the differences in the environmental cost of each, we only show one path and display the environmental data above it. This change was due to the same reason as the first change. The CRUD functionalities remain pretty similar, except for being able to choose from different importer options.

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

Our application achieved its goal of showing the environmental impact involved in producing and transporting the food on your plate, based on where you are located. Users can learn a lot about the actual cost of their purchases, and this can help them make more intentional decisions based on environmental impact. We were also able to implement the grocery list feature, which is a useful component for users to use more practically. By allowing users to create grocery lists with different product combinations, they are able to see the differences in their carbon footprint and fuel consumption. This feature will also help users make greener grocery choices.

On the other hand, the setup of our calculations is based on some level of estimation, which makes some of the results a little less accurate than we would like them to be. For example, we calculate fuel consumption based on the fuel needed for a plane to fly from the product location to the user location. This may not be the most accurate representation of fuel usage because products are transported in vehicles other than planes. To combat this, we can source more relevant data or collect data on our own.

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

For the most part, we used the same data sources that we outlined in Part 1. We did have to clean a lot of the data, so our relations in GCP have less data due to duplicates and irrelevant information that we removed from the sources. We also generated some of the data in the environmental cost table due to not having the required 1,000 rows from Stage 3.

Discuss what you changed 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?
userData:
We did not change the implementation of the user table since creating the ER diagram.
productData:
We added EC_Id and location name as attributes in the product table. EC_Id was added to help us map the environmental cost of each food group to the product names in the product table. The location name was added primarily for back-end readability, so that we could easily see where the products were sourced from when cleaning the data and writing queries.
locationData:
We did not change the implementation of the location table since creating the ER diagram.
environmentalCost:
We did not change the implementation of the environmental cost table since creating the ER diagram.
groceryProduct:
We added LocationId and glId as attributes in the grocery product table. LocationId was added to store the locations that each individual grocery item is being sourced from, allowing us to reference it after the item is added and to calculate fuel usage and total emissions accurately. glId was added so that users can have multiple grocery lists. This makes it easier for users to compare the environmental costs of different grocery list combinations.

Discuss what functionalities you added or removed. Why?

We added the functionality to have multiple lists because we thought it would make it easier for users to compare their grocery lists. We removed the functionality of having multiple

importer options because we did not have sufficient data to implement that feature. We removed having a hover pop-up on the map visualization as we already display that data above the map anyway. We added functionalities to duplicate lists and move items from one list to another in order to make user interaction easier. For example, if a user only wanted to change a few items between list 1 and list 2, they could easily duplicate list 1 and change those items rather than creating an entirely new list.
Explain how you think your advanced database programs complement your application.

I think that the advanced database programs added a lot to the functionality of our application. The trigger we implemented allows users to start off with a basic grocery list comprised of bread, eggs, and milk, with a default location. This is useful because users are able to see what a grocery list looks like, which may make it easier for them to understand how the lists part of the application works. Since users can remove items from their list at any time, those who opt into the default grocery list can always remove individual items or delete the list entirely.

We used two advanced stored procedures for this application: one to allow users to duplicate their grocery list and another to display the current grocery list with advanced calculations. The procedure for duplicating a list complements the application because it makes it easy for users to compare lists that may have only one or two differences and see which overall cost is lower. The procedure that displays the list and calculates total costs is integral to our application, as it shows users the fuel usage and total cost for each item, as well as the total cost of the entire list. The stored procedure calculates the distance between the user's location and the product's location, taking into account the curvature of the Earth, and calculates fuel usage based on the average fuel consumed by a plane per mile. It also multiplies the total environmental cost based on the quantity of each item.

We also used two advanced transactions for this application: one to search and output environmental statistics for individual items, and another to move an item from one grocery list to another. The transaction for search and output is integral to our application because it forms the backend of our home page. This transaction also calculates fuel consumption on an item-by-item basis and displays a more detailed environmental impact statistic. It is implemented as a transaction in case any of the steps, such as getting the user's location, fail. The other transaction enhances the application by saving users the trouble of deleting an item, switching to another grocery list, and then re-adding that item. By reducing two steps down to one, this transaction improves the user experience. It is implemented as a transaction to ensure that the movement of the item from one list to another does not create any inconsistencies.

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.

Kathy: We had trouble dealing with different versions of Node.js and Next.js across our team members' local environments. Because we had slightly different versions installed, our package.json and package-lock.json files kept conflicting and breaking the app when we pushed and pulled from GitHub. We learned that it's really important at the beginning of the project to standardize the version of Node.js and Next.js that everyone is using.

Kevin: We also struggled with displaying user-specific information after login, such as showing the correct profile info and grocery list for the person signed in. It was tricky to keep track of which user was currently active, especially when switching pages or refreshing. We learned that we needed to manage session state carefully on the frontend and backend, using things like cookies or local storage to keep track of the authenticated user.

Medha: One technical challenge that we struggled with was figuring out the database connection, specifically the IP network connection. We did not realize that we had to individually add each of our IPs to be able to connect to the backend on GCP locally. We also did not realize that your IP changes based on the location you are in and the day of it is. We learned that we need to update our IP addresses in the connections section of GCP each time we move to a new location or work on our project on a different day.

Vani: Another technical challenge that we struggled with was setting up user authentication and building the password and account manager. At first, we weren't sure how to safely store user information and connect it properly to our database. We realized that we needed to create a separate users table and store hashed passwords instead of plain text for security. It also took some time to figure out how to set up our backend routes so that the signup and login processes correctly referenced the database to authenticate users.

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

Everything that we changed has been covered through the other questions in this write-up.

Describe future work that you think, other than the interface, that the application can improve on.
I think sourcing more robust databases with more relevant information could significantly improve the accuracy of this application. For example, having environmental cost data from the last five years and updating it as environmental costs change would make the numbers more accurate for users to reference. Sourcing relevant importer data would also be impactful, as it would allow us to implement the original design of showing users different importers from various companies and locations to help them make informed decisions. For example, a user could simply search "banana" and be presented with three to four importer options that import to their location, along with specific environmental cost data for each importer. This application design is definitely outside the scope of what we could have implemented for this class, due to the sheer amount of specific data that would be required. It would also be easier to implement if the application focused on a specific food group rather than allowing any item. Additionally, it would be beneficial to have more variety in the products that users can search, as our current implementation focuses more on food groups than specific items (ex: guava instead of fresh fruit).

Describe the final division of labor and how well you managed teamwork.
Our team collaborated closely throughout the project, either by meeting up to work together or by dividing the work evenly and making individual progress. While everyone contributed to all major aspects of the application, each member took the lead on specific areas:
Kathy: Wrote SQL queries and implemented advanced database components such as transactions and stored procedures, and contributed to the product keyword search functionality. Wrote transaction and backend for moving a product from one list to another.
Kevin: Built the dynamic map visualization for the homepage, developed front-end components for the homepage, and modified backend API routes to support the map features. Wrote the API routes for the location function and the get user location function.
Medha: Wrote SQL queries and developed advanced database features, including trigger, transactions, and stored procedures, built the front end for the grocery lists page, and created backend API routes to support grocery list operations.
Vani: Focused on developing the front end for the sign-in and sign-up pages, created several backend API routes, and helped integrate user authentication workflows. Also developed hashing and cookie management systems to allow users to access profile-specific information.