CS 411 Final Report: BizBites

Changes in the direction of the Project

There were no changes in the direction of our project from the completion of the project from our initial proposal. We only had some issues implementing some of the bolder ideas that we had when drafting up the project like GPT integration and having all different user interfaces for each portal to our website, but overall we stayed fairly close to our initial goal.

Achievements and Failures Regarding Usefulness

The team successfully connected businesses with restaurants for tailored deals and created multiple UIs to enhance accessibility for employees, businesses, and restaurants. Lastly, features like order tracking, dietary preference management, deal analysis, and AI-powered meal recommendations were implemented. At this current moment, the application does not fail in any aspect to deliver the specified goals.

Changes in Schema or Data of the Application

There were no schema or data source changes for our originally proposed application. We are still using the original database with the same parameters and tables that we initially had when setting them up for this project.

Changes in ER Diagram or Table Implementations

The Deal entity was removed as a standalone entity with its unique identifier (D-Id), as this was unnecessary since Deals were tied between the Business and Restaurant elements of the schema. By integrating Deals into the association between the two entities after the change in our schema, it eliminated redundancy and was associated in a relevant context without the need for its own identity.

Added or Removed Functionalities

The new functionalities added to the application were transactions and filtering. The first transaction is meant to identify a restaurant's high-value meal deals. It calculates the average discount percentage, counts the distinct menu categories, and computes the average price for items within the restaurant. An example would be filtering restaurants where the discount is larger than 30 percent discount. This feature will allow restaurants to maximize their participation with local businesses. The next transaction evaluates businesses based on the number of deals and average discount percentages they have with restaurants. It also identifies premium deals that exceed say 15 percent. The query ranks businesses with above-average

discounts compared to the global deal average. The transaction allows insights for businesses to negotiate deals with restaurants, ensuring better partnerships.

The first filtering feature examines food categories based on nutritional and pricing metrics. It computes average calories, protein, and item counts for foods in each category and only displays the food items that match the filter. The second filtering feature identifies restaurants offering healthy food options. It calculates metrics based on protein-to-price ratios for each restaurant. It filters food items, for example with protein greater than zero and a price below 1.5 times the average price of all food items. Only restaurants with at least two food items meeting this criterion are shown.

An employee profile was added to demonstrate the CRUD operations. There is an employee account creation page where the employee can create an account by filling in their name, phone number, company, dietary restrictions, etc. This will add the new employee to the DB. Once logged, a token is created to save the employee's data. The employee can then modify any information should they need to change anything. This will also update the DB with the respective results. Additionally, the employee can delete the account and this will be reflected in the DB.

Teams Technical Challenges

The team encountered a merging issue and lost some progress. The team has issues with the use of tokens within the project. Generating the token from GCP required that the token was securely issued. Displaying the token in the web console required accurate configuration of APIs. Using the token in the front end led to issues because of improper handling of secure storage. Linking these steps took a lot of testing to ensure that the token was implemented correctly. We also had an issue where we lost the most recent commit in our version control which caused us to lose a few hours of work and struggle to find out what parts we were missing as everyone had a different version of the code that didn't work with the others.

Juan Diego: When uploading CSV files into the database, the entire file was uploaded in one cell. The solution to this issue was that some of our columns were all of the same type and the database interpreted the CSV file as one cell. By changing the columns to the correct data type, the database was forced to type-check and ensure that the CSV rows were split into the appropriate fields.

Calvin: It was recommended that in the beginning, we develop the back end to set up the routing before creating the front end; however, we did the opposite. This caused a lot of extra work trying to fix what was first implemented on the front end. This also caused some of the queries to not parse correctly and led to extra time trying to figure out the discrepancies.

Nick: As someone learning React by trial and error while building the end for a few web pages, it was overwhelming at first. Breaking down the design into reusable components is confusing, and I often find myself duplicating code instead of reusing it. Setting up navigation with react-router-dom was another hurdle. I kept getting errors because I forgot to wrap the app in a <BrowserRouter>. Fetching data from APIs seemed simple, but I ran into issues with loading states and figuring out where to handle errors.

Hrishi: I had some challenges working on getting the macros to match between the food stats table and the meals. It was mainly due to the discrepancy in how the data was fetched from the backend. The meal data and macro information were coming from different API endpoints, so there was a need to synchronize this data by matching the meal name with its nutritional values. For integrating GPT implementation into the site, I utilized the OpenAI API to allow users to ask for meal suggestions or other food-related recommendations directly through the UI. Integrating GPT into the site was a technical challenge primarily due to the complexity of managing API calls and ensuring smooth communication between the frontend, backend, and OpenAI's API.

Conclusions and Future Work

In conclusion, there were no changes from the start of the project to the submission of the final project. With the inclusion of the ChatGPT chatbot, its current function will give recommendations based on the API but further implementation will be needed to query the database and update the web pages. Another thing to consider for future work would be to add images of all meals, food items, and company logos. The way the meals are presented is in a horizontal scroller, in the future, it would be nice to have it displayed in a nicer format. We would also like to optimize the search bar to query the database and give suggestions instead of waiting for a completed response when giving meal suggestions. We would also like to make the GPT implementation more tailored to the data shown on the UI and better trained to give more personalized and useful data to the customer while querying the database for accurate info.

Division of Work

Calvin: Back-end integration and development
Juan Diego: Back-end integration and development
Hrishi: UI design, GPT Implementation
Nick: UI Design, search bar, Final Report