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).
2. Discuss what you think your application achieved or failed to achieve regarding its usefulness.
3. Discuss if you changed the schema or source of the data for your application
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?
5. Discuss what functionalities you added or removed. Why?
6. Explain how you think your advanced database programs complement your application.
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.
8. Are there other things that changed comparing the final application with the original proposal?
9. Describe future work that you think, other than the interface, that the application can improve on
10. Describe the final division of labor and how well you managed teamwork.

1. Our final project followed through on the initial direction and path we chose for the project during the stage 1 proposal submission. Our app was designed to help people track their calories, their activities, their nutrition consumption, and other factors that impact their day-to-day lifestyle. We had certain kinks in mind like a leaderboard and a food suggestion system. There were minor tweaks along the way but it didn't divert our original direction and delivered on the said outline of our application. Our timeline matched with the expected timeline for the project and we felt that our project progressed quite consistently through the stages.

2. We have made it possible through a large food dataset for people to track their diet, with a huge amount of variety of food available on the app. Thus, people tracking their consumption of certain nutrients in these food items has been made available for use. We failed to, however, incorporate alerts when the intake of, let's say cholesterol, rises beyond a certain threshold. This application, through the virtue of diet tracking, also helps people who generally want to maintain a balanced diet and track their calorie intake for a particular fitness goal. Paired with the activity tracker, people can successfully track their fitness goals holistically. The activity tracker has allowed us to produce two special parts to our application : an activity leaderboard and an activity-based food suggestion system.

3.
Indeed, several modifications were made to the schema and data sources of our application throughout its development. In the initial stages, we recognized the importance of enhancing security for user sign-in processes. As a result, we introduced a password column to the user table during stage 4, ensuring a more secure authentication mechanism.

Another change involved the 'Activities' table, where we incorporated a FOREIGN KEY constraint named "userid" as part of the Primary Key. This alteration established a clear relationship between the 'Activities' table and the corresponding user.

To maintain data integrity and ensure a clean database, we implemented the "ON DELETE CASCADE" functionality in the 'Activity', 'Health', and 'Goals' tables. Since these tables are considered weak entities and depend on the 'userId', applying this constraint allowed for automatic deletion of associated records when a user's account was deleted. This approach contributed to a more streamlined and organized database structure.

During stage 5, we introduced the 'Favorites' and 'Consumed' tables. These relational tables were established to establish connections between users and food items, enabling more efficient and effective functioning of our application.

These alterations to the schema and data sources were made strategically to improve the security, integrity, and overall performance of our application.

4. We made very few changes to our original ER diagram and table implementations. Our initial implementation would have the Health of each user updated for each of their Activities; however, due to time constraints we were unable to develop a procedure for this functionality. Otherwise, we would have a stored procedure triggered by the creation of an Activity to update the Health table. Another difference is the Favorite relation. We had initially thought of making food suggestions to users based on the similarities in favorite foods that they had with other users. Instead, we chose to make suggestions based on calorie goals from the Goals table and calories burned from the Health/Activities tables. We made this change because we want to promote a healthy and balanced lifestyle. While tasty foods can be appealing, it defeats part of the purpose of the app. Additionally, users using BodyWatch likely have the mindset of monitoring and maintaining a healthy lifestyle. We want our app to serve our users, so we shifted toward suggesting users foods that they can eat within the goals they set for themselves while considering the activity that they had done. We feel that this is a more suitable design which can be further enhanced by having Activities update the user’s Health.

5. We implemented the functionality of a food suggestion as stated above. We also added the functionality of a leaderboard and percentage of user’s beaten in terms of calories burned. The leaderboard simulates a friendly competition among our users to motivate them to exercise more. Similarly, we have the functionality of user’s knowing what percentage of user’s they beat.

The comparison metric here is how much a user has worked towards or surpassed their goal. We eliminated the functionality of user's being able to Favorite foods due to the reasoning stated in (4); however, we do foresee many use cases for this functionality in terms of food suggestions and optimizing food searches.

6. Our advanced database program was a stored procedure and a trigger. We used a stored procedure to calculate what percentile of the total number of users each person is with regards to completing their goals. This would happen whenever a user logs an Activity. This complements our application well because it opens the door to an app that can host competitions amongst a group of users, i.e. a user's friend group. It can also be motivational in the sense that user's can strive for higher percentages and thus they would ideally set attainable yet challenging goals. Our trigger limits users from creating a goal of over 3000 calories. This is because we do not want users to mistakenly set an unachievable goal for themselves and have their statistical record skewed by faulty data. The trigger can be overridden by updating a goal to have more than 3000 calories, so users who do want to set high goals still can.

7.
Shiqi:
During the development of our application, the team encountered a notable technical challenge related to the implementation of the delete method. In the backend, I utilized the "@Body" annotation, which required passing a JSON body in this particular API. However, in the Android application, we used Retrofit to handle API requests, and we discovered that Retrofit does not support passing a JSON body in the DELETE method by default.

To overcome this obstacle, I conducted thorough research and found a workaround. I learned that we could use the `@HTTP` annotation with the parameters `method = "DELETE"`, `path = "health"`, and `hasBody = true`. This replacement allowed us to successfully send the JSON body within the DELETE request using Retrofit.

For future teams embarking on a similar project or those maintaining our application, I would advise them to keep this workaround in mind if they encounter a similar issue with Retrofit and the DELETE method. Utilizing the `@HTTP` annotation with the specified parameters can effectively address the challenge and enable the passing of a JSON body in DELETE requests within the Retrofit framework.

Manit:
The team encountered multiple technical challenges working with the food database. The database is large and requires a significant amount of data cleaning. We initially used Python to format and do a high level clean up of the data; however, we were still left with many NULL entries or inconsistencies in data. One issue, in particular, was how we handled the serving size of the food. The dataset we used had two columns for a serving size with one being NULL when the other had a value. Additionally, the columns were in different units of measure. We addressed this by enhancing our queries to exclude NULL or 0 values, and we used an IFNULL

statement to handle the two columns of serving size. We had spent significant time debugging our advanced queries because of messy data, so I would recommend future groups to write queries that handle all edge cases and account for dirty data entries, even after the initial data cleaning process.

Tanishq:

One of the main issues we had, as Manit mentioned above, was related to the data being dirty with Null values and simply garbage at some points. We definitely had a hard time collecting clean data, which encouraged us to shift to generating data for the purposes of the MVP. We used Mockaroo to generate this data. Activity data was supposed to be initially extracted from the Apple Health API but due to the complexity of parsing the data, we decided generating reliable and normal datasets would serve us better. This could be an issue in any data driven application and such solutions can serve as future reminders for developers requiring some data to showcase their application.

8.
As a result of time constraints, certain modifications were made to the final application compared to the original proposal. The implementation of Favorites and Consumed tables in both the backend and frontend had to be omitted. Similarly, the addition of a password column to the user table was not included in the final version.

Regarding creative functions, instead of implementing machine learning functions as initially proposed, a bar chart was incorporated into the leaderboard table.

These adjustments were made in response to the limitations imposed by time availability, ultimately resulting in a modified scope for the final application compared to the original proposal.

9.
Firstly, within our database, we need to add a password column to the user table. Furthermore, we should implement user Favorites and Consumed tables in both the backend and frontend.

Subsequently, in the realm of the frontend, it is of utmost importance to prioritize the security of user data, particularly pertaining to their personal information.

Lastly, in the backend realm, our current implementation utilizes a local host configuration. To enhance the standardization and security of the API, we consider upgrading to the HTTPS protocol.

10.
Shiqi Implemented front end: create an Android application; receiving data from backend's api and display to user;
Implemented back end: using spring boot connect with gcp and created the apis;

In addition, implemented creative functions and triggers to enhance the system's functionality.

Manit specialized in advanced queries, stored procedure implementation, and logical design. Assisting Tanishq with implementing and maintaining the database.

Tanishq was responsible for the database implementation and maintenance on GCP. Set up and managing the database infrastructure

Overall, the team demonstrated effective collaboration and coordination in dividing the workload based on each member's expertise. The division of labor allowed for efficient progress in different areas of the project, showcasing a well-managed teamwork approach.