1. In stage 1 we stated that the users will be able to connect with other users, and be able to see what kind of workouts, amount of calories, and favorite foods their friends have. But unfortunately due to time constraints we were unable to implement this feature.
2. The main components that we were striving for with our application was first that the user would be able to track their calories, make a workout program, and be able to connect with other users. Users were able to track their calories and add desired workouts to a workout split that they could use. So our application was able to achieve these components regarding its usability. One thing we wanted our program to be able to do was to be able to connect with other users to add some sort of social component. We were able to allow users to check if they shared any favorite foods and for them to see if they would be able to go out for food with other users, but ultimately we wanted more connectivity between customers.
3. We slightly changed our schema in our database. One change that we made was we added the exercises within our database so that it could relate to the workouts database that we had. Additionally, we changed some of the relationships within our database so that it could make more sense, such as making Customers have a 1 to many relationship to food (which originally started off with a 0 to many relationship to food), and making Split have a 1 to 1 relationship to Customers. We did not change the source of data of our application in any way.
4. We originally started with 4 tables, Food, Splits, Workouts, and Customer. When we started to create primary and foreign keys we realized that there was no way to connect Workouts to any table, after contemplating for a while we decided to create a new table Equipments to basically link Workouts to. As we started to implement the tables we realized our relationships such as, one-many, many-many, one-one, etc were slightly off, so we had to fix those relationships as well. I think this design is more suitable because no one table is overloaded with information and there are no functional redundancies in our schemas, also the way we split our information makes it easy to navigate for anyone.
5. Functionalities that we added to our application were mainly social aspects to the app. For example, we integrated one of our advanced queries that recognized if multiple users have the same split with each other. Another aspect that we added was a stored procedure that compared two users' calories left and lists all the foods that they can eat.
6. The first advanced database program is a trigger that we implemented that adds a food to our food database if it isn't already a present when a user inputs their favorite food. This complements our application as we want to make our food database as comprehensive as possible. By adding new foods to our database upon a user input we can ensure that it is constantly getting updated. Our second advanced database program is a stored procedure that compares two users and the amount of calories they have remaining and the foods that the users can eat with the total calories under their limit and see if they can go together and eat some of these foods. For example, say there's a User 1 has 500 calories left and that a User 2 has 400 calories left. Our stored procedure would find foods that both User 1 and User 2 can eat together. This complements our database program because we are striving to add a social component,

which is something that our stored procedure is able to do by encouraging users to connect and hit their calorie goals together.

7. One challenge Aryan faced was integrating the frontend with the backend. Since our backend was in python and frontend was in HTML/CSS and basic JS we had to use flask to integrate the front and back. So learning a new technology from scratch was a very challenging task. Nikil faced troubleshooting issues with setting up the localhost database, as he had no prior knowledge in sql workbench. While implementing the frontend, Neil faced the issue of creating a new design from scratch while keeping constraints. He had to be creative on how to create a user-friendly interface for all the different routes and make sure each button and back click is redirected correctly. While creating the trigger Ayan ran into the issue that the Food table was not updating and functioning properly, so he had to debug his trigger and recreate in such a way that the Food table functionality doesn't get disrupted.
8. Some things that changed from our original proposal are with both the User Interface and our functionality. The first change had to do with the way our User Interface was structured, as before we planned to cram everything with our workouts, splits, equipment, and foods. However, this was very clunky and awkward with the way all the information was crammed in one screen. Therefore, the UI change we decided to make was with implementing a page for each database associated in our application. This made the UI a lot more user friendly and readable. The other change that we implemented was with our functionality. We decided to implement our stored procedure to be part of our application in a creative sense. It checked if two users have calories left over, and checked the foods that they could both eat together.
9. Future work that our application can improve on is adding a lot more of a social aspect to it. Many workout trackers are individual based, but we were working on adding social aspects to this workout tracker. This is seen with the stored procedure that we implemented together in the application, and future work can expand on this. For example, a feature we could add is the ability to add other users as friends on the application, so that they can track their progress together and see if they have similarities in their workout, eating habits, or goals. Another expansion is adding a point system that encourages friends to challenge each other.
10. Aryan and Neil were mainly in charge of the front-end and integrating the application with the backend. On the other hand Nikil and Ayan were in charge with coming up with the SQL queries, the advanced database programs, and setting up the backend of the application. We would all work together when completing the work to ensure that there weren’t any work imbalances and so that we could set aside time to lock in on finishing up the stages. We managed the teamwork really well and our happy with how the group project turned out.