1. Please list out changes in directions of your project if the final project is different from your original proposal (based on your stage 1 proposal submission).

Overall, the final application implemented by our group is in line with what was envisioned at the beginning. We tried to implement an application to help users, especially UIUC students, to know the inventory of a certain item in major supermarkets in Champaign. This is to alleviate the time students spend shopping at major supermarkets, especially during the school year when demand for daily necessities is high and inventory is slow to replenish.

In our project proposal for Stage 1, we presented two ideas for our project: First, to implement a crowdfunding system where any user can update the inventory of an item in a particular store on the website; Second, to allow users to compare the price of an item in each store in Champaign, based on the price provided on the website. Both of these ideas were implemented in our application.

So we can say that the direction of our project hasn't changed. However, during the development process, we improved some details of functionality and implementation. For example, in the project proposal, we mentioned that the user could group by the info of a product in a given time period, but we finally eliminated this feature and simply showed the status updates uploaded by the user in reverse order. We believe that the latest status update has more reference value and weight than the older data. In addition, we were not able to develop the Rewards Point System and visualize the historical and seasonal pattern due to time constraints, but we still think these parts are valuable. We are considering keeping this part of the implementation for our subsequent work.

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

According to our original plan, we implemented the main function which is to provide a platform for UIUC students to have easy access to the stock situation of products in different markets. This feature considerably saves students’ time for shopping, especially during the back-to-school period.

In particular, we have built multiple navigations for users to reach the stock status and users can find it by specifying the product category or the market name. We also provided several interfaces for multiple roles to interact with the data, for example, the customers can use the search bar to retrieve the retail data, create a status record for stock situations and give likes to the status provided by other users, besides the administrator can update the retail price and even delete the whole record. Moreover, our application can support some special features like applying discounts on products or some advanced queries including getting the number of users for every number of likes for Costco and Walmart together. Besides, we integrated advanced storage procedures and triggers with useful and reasonable advanced queries.

However, we failed to figure out a better way to incorporate a few of the advanced queries into our application and we just temporarily left them as a demo, so our users cannot use these features easily. We also failed to deploy a Web crawler in our system to automatically retrieve data and store it in our database. Due to the limited time, we are unable to implement some creative features including giving users reward points for a certain amount of contribution or providing visualization of demands and supplies of products.

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

We change the source of the data. Initially, we plan to use a web crawler to obtain product information from different markets' websites. However, it is difficult to match the same product sold in different markets. Therefore, we collected product information including name, price, category name, and subcategory name from a single market and allocated them randomly to different markets with some changes in their prices.

For other parts of the data, we follow our original ideas. We generate users with different names, emails, and passwords. We generate the Status of the comments based on the existing Retails and Users Table.

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?

Our group's database implementation has minor changes from the original UML design.

In the original design, we planned to use INT for all the IDs (Keys) of the table but the CHAR(32) is applied in the real implementation. We believe it is a more suitable design for three reasons. First, the CHAR(32) ID could be generated separately without referring to other data, and a data record can have the keys before being inserted into the database, which makes the data generation and programming easier. Second, the data range of INT is relatively insufficient for the size of our dataset. Third, although the CHAR(32) will slow down the sorting of the keys compared to INT, there are no queries that require sorting on the keys in our application. Fourth, it fits better with the project architecture with many split tables, and it makes it easier to make changes to the associated table data when integrating systems.

Meanwhile, we corrected a small typo in the Stage 2 database design. As envisioned, we have three entities, Markets, Products, and Retails, where the Markets table and Retails table, Products table, and Retails table both have a one-to-many relationship. Theoretically, what we need to do is to use the primary key of the Markets table and the Products table as the foreign key of the Retails table. But in the Stage 2 database schema, we accidentally use the primary key retailID of Retails table as the foreign key in the Markets table. We corrected this small mistake in the implementation.

Besides, we removed the Brand table from Stage 1 in Stage 2 because it is relatively laborious to generate the brand information, and the brands are already contained in the product name, which is sufficient for users to know. However, in the further, the Brand table could also be added to make the application more useful.

5. Discuss what functionalities you added or removed. Why?

Generally, we add 3 functionalities:

1. We added a dedicated admin page for administrators of this application to manage the retail information, and status information because as an application designer, we cannot grant update and delete access for customers, so we have to create an administrator role.
2. We added a feature to apply discounts according to predefined rules on the products, and users can directly see the page showing the discount information. It could make our application more maintainable by utilizing some advanced database programs.
3. For a more user-friendly interface, we designed and implemented some useful UI elements like a discount information card, multi-level menu, and popup modal for updating the price. For a better user experience, our website also supports lively reload.

And due to the limited time, we remove

1. the feature of sorting and grouping the retail status by the created time
2. the feature of giving users reward points for a certain amount of contribution
3. the feature of providing visualization of demands and supplies of products
4. the feature of recommending products displayed to users by their preferences

6. Explain how you think your advanced database programs complement your application.

Generally, we use three dedicated advanced queries and a storage procedure, and a trigger.
The advanced queries allow our programs possible to aggregate and retrieve information in one go.

For the storage procedure, it will calculate the discount for all the products based on the predefined rules. It involves complex processes and will be called many times. These complex processes could not be written in a single query. The storage procedure only returns information to the back-end server once, which is more efficient and maintainable. Meanwhile, if we want to change the discount rules we only need to change this storage procedure rather than changing the raw queries written in the back-end JAVA program and restarting the whole service. Therefore, the discount information could be updated in real-time while obeying the ACID of the database.

For the trigger, when updating or inserting values to the Likes Table, the Status table will be updated accordingly. To be more specific, we have values numOfLikes and numOfDislikes stored in the Status table. When the user updates their preference to a status, numOfLikes, and numOfDislikes of that status should be also updated.

It is important because it helps to synchronize the updates and deletes on the User table to the Status table. It is inefficient if the program loop over all the users to calculate the numOfLikes and numOfDislikes. Meanwhile, using the back-end server with multiple queries to modify the two tables instead of the trigger is difficult, inefficient, and less maintainable and the two tables are likely to be inconsistent because of the bugs in back-end queries. The trigger automatically syncs the updates and deletes and the back-end server only needs to access and modify the user table, which makes the database consistent.

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.

The first is about the realness of the data. We planned to use real data from our nearby supermarkets, as we tried to crawl data from the official websites of Walmart, Target, and Costco. However, even though we strictly followed the restriction stated by robots.txt, our web crawler was still blocked by those websites. As a result, we generated data by combing an open-source database and also we randomly generated the remaining missing data.

Another one we encountered in the process was indexing. We expected our indexing design to greatly improve the performance of the database, but we found out that the performance was already at its optimal so whatever we do to the database design would make the execution time even longer. The main reason for the problem is due to the built-in design of the MySQL database, which automatically treats the primary keys of the table as the index.

We also struggled a lot with the advanced queries. We made a lot of efforts to come up with reasonably advanced queries and integrate them into our web applications. So we brainstormed a lot of scenarios based on our real-life experiences. As a result, we redesigned and adjusted some parts of our web application in order to integrate those advanced functions. Hopefully the next time we can design those functionalities first and then start the implementation process.

The fourth difficulty is our application performance. It can be observed that when the request is frequent, the network performance of the website is poor due to the frequent request time. We tried to solve the issue as our further study by adding a web cache to the application, resulting in less overall delay.

The fifth challenge is the front-end, back-end, and database integration. At the beginning of the project, we did not familiar with the protocols and approaches to connect them, and the protocols between the subsystems are not clearly defined at the beginning, Thus, we spent a significant amount of time trying to send correct packages between them. In the further, we will learn, design, and test protocols first before starting to implement the program.

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

Compared with the original proposal, the UI layout of our application is much more different because the UI mock-up in our original proposal is just our initial ideas which do not contain all the features we want to provide for our users. Considering the functionalities we want to implement and the number of database tables we have, we decided to build multiple pages and the needed data should be requested every time a user accesses a new page.

9. Describe future work that you think, other than the interface, that the application can improve on.

There are several aspects that could be improved in the future.

- The structure of the system could be reorganized and the subsystems should be separated clearly.
- The frequently accessed queries should be further optimized by identifying key indexes, and the database tables could be further decomposed.
- The database design could be further improved by adding referential integrity constraints
- We could introduce Redis to accelerate the high concurrent requests like rating and commenting, and sync with MySQL periodically. Besides, Redis is perfect for session storage and management, reading and writing will be fast.
- Neo4j could be used to replace some parts of our database design where the relations between entities are complex. The markets-retails-products entity and relation could be replaced by a graph.
- Add the feature of sorting and grouping the retail status by the created time
- Introducing reward points systems for users with a certain amount of contribution
- Providing visualization of demands and supplies of products
- Recommending products displayed to users by their preferences

10. Describe the final division of labor and how well you managed teamwork.

Every member of our team has great teamwork skills. We have all devoted a lot of our time and effort to this project and all of us have had a pleasant experience working together. To ensure every member participated in the project, we held a meeting every time when we need to make a decision or brainstorm project ideas. If the time is allowed, we also do the coding together which makes us more focused and thus improves our efficiency. More importantly, the key to our successful teamwork is our sense of responsibility. We have a clear division of labor and all of us know the importance of being a reliable teammates and delivering the work timely. From a technical perspective, we learn to make good use of Git. Apart from some basic commands like git add/commit/push, we work on different branches for this project which is a good practice for avoiding interfering with each other.

The detailed final division of labor is as follows:

1. Database design and implementation
   All team members participate in this part and make an equal contribution.
2. Front-end dev
   Ziyi Chen and Yitian Hu utilize React with Typescript and MUI to build a user-friendly front-end project.
3. Back-end dev
   Mike Wu and Zeyu Wang utilize Java Spring Boot and REST API to build an efficient service for the project.

4. DevOps
   Zeyu Wang deploys our entire project on GCP and ensures it always running correctly.
5. Documentation
   All team members participate in this part and make an equal contribution.