CS411 Stage 6 Project Report
Group 11

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).**

So far, we haven't changed our general direction. Throughout the project process, we use the stage 1 proposal as guidelines. But different from the UI mockup the proposal has listed, we have some detailed features like displaying product photos that haven't been implemented yet. The defect does not affect the functions required by the project, but we will make up for them in future development. In addition, due to time constraints, we only developed functions related to merchants in the first stage, and we finished developing the rest functionalities related to customers in the second stage.

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

Regarding achievements, we have fulfilled the following functions: Firstly, our application has implemented the functionality of listing new products and making information changes to existing products; Secondly, the application achieves the functionality of automatically generating sales reports and regular customer reports according to the sales of each product; Thirdly, functionalities that are available for customers, including querying products, adding products to the shopping cart, editing shopping cart products, and placing orders have been implemented. This series' functions are defined through triggering and transaction setting up. Finally, we additionally create a 'special offer' algorithm, that can give target customers unparalleled promotion code based on a trigger and stored procedure, which we will detailedly introduce in *question 3*.

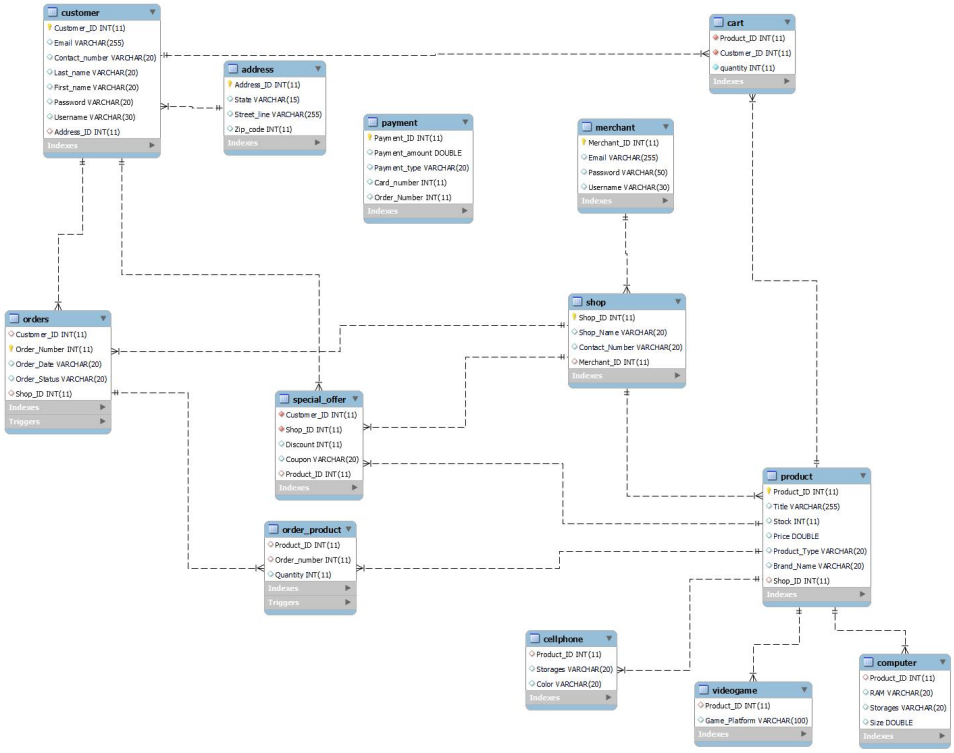
3. **Discuss if you changed the schema or source of the data for your application**

We modified two places based on our application schema:
First, we re-edited the *Order* table. We originally expected that the relationship between Order and Shop is one-to-many, that customers may buy multiple products of different brands at one time, but that conflicts with the one-to-one relationship between Order and Payment. Therefore, we adjusted to only have one item per order, which makes it a one-to-one relationship to generate payment info for each shop.
Second, we innovated a Special Offer table based on the record of Payment and Order in our database, which finds the most expensive product each customer has purchased, and we will send them discount offers that are available for customers purchasing the same products next time. The discount rate is calculated through a stored procedure, which returns the total purchase amount for each customer who has spent over 1000 dollars on our application. Customers whose total purchase values are over 10,000 will receive a

30% discount. Customers who spend over 8,000 dollars will receive 20% off, and customers who spend over 5,000 dollars will receive 10% off.

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?**

Compared with the original design, the final design has three minor differences: first, some unnecessary tables are removed, like the *Image* table. Due to the time limitation, we didn't add images to our website, so we removed the empty *Image* table. Second, the relationship between Order and Shop is changed from one-to-many to one-to-one. This has been mentioned in question 3, we changed the relationship type to generate payment info for each shop. Lastly, a Special_Offer table has been added in the final design, to make sure our trigger and stored procedure work.



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

Compared to the initial plan, we have added one functionality and removed one. We have added the special offer functionality, which is implemented with a trigger. We have described its functionality in detail in the previous part. The special offer can help merchants automatically match discounts to customers based on historical spending amounts without manually checking. This saves the merchant time and ensures the accuracy of the data. In addition, we removed the filter which can filter items by price

and popularity. This is because our current shopping functionalities already meet the needs of customers looking for products. Also, in electronic supplies sales, few customers choose to select products based on price ranking.

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

In our project, we implemented transaction, trigger, and stored procedures. The transaction helps us to maintain consistency so that if a single operation fails we can roll back and not modify the data. The isolation level we choose is read committed, which ensures that the same item will not be ordered repeatedly. For the trigger and stored procedure, we implemented the function of special offer, which helps merchants to automatically match discounts to customers based on historical spending amounts, thus reducing the frequency of data updates.

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.**

- Weiru Sun

One challenge in our project is user login functionality. The user login functionality of Django JWT does not match our database model, so we used the Django rest frame instead. In detail, we use the get functionality to implement login, post functionality to implement register, and put functionality to implement update information.

- Runyu Liu

Passing parameters (product_Id, customer_id, merchant_id, order_id) between web pages is required. We are inspired by google drive's URL and save the information in the URL so that the information of the user, product, and order will not be lost when switching web pages.

- Haoyu Zhang

Debugging is a frequent problem for our team. If the API cannot work at the front end, we need to figure out where the problem is. Therefore, we need to test the functions from the database (SQL), to the backend (API), and finally to the front by using the unit test.

- Yizhan Xue

The transaction is one of the challenges we encounter in our project. For database transactions in Django, the default behavior is to run in auto-commit mode. We need to separate one raw SQL into multiple statements to return the error and roll if there is something wrong with any step of the transaction.

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

Our final project implementation was highly compatible with the initial plan, and our implementations of functionalities were in line with expectations. We also made unplanned improvements, such as the special offer and transaction isolation levels mentioned in the previous section. In summary, we have completed the expected plan and made some improvements.

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

For the improvement, we will combine JWT (JSON Web Token) with the log-in function to improve safety in the future. When a user logs in, the authentication server verifies the credentials and issues a token signed using either a secret salt or a private key, and the User's Client uses the token to access protected resources by passing it in the HTTP Authorization header.

Also in the future, we will make improvements to application aesthetics, like adding more product images on the web.

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

Under the leadership of our team leader Runyu Liu, we divided the teamwork fairly. For the database and data, Weiru Sun, Haoyu Zhang, and Yizhan Xue are responsible for collecting and generating data, and Runyu Liu is responsible for building the database on the MySql platform. For the development, Weiru Sun was responsible for the application's home page, login/register page, account page, customer's product overview page, and product detail page. Runyu Liu was responsible for the customer’s cart page, order submission page, purchase history page, order status page, and seller’s start page. Haoyu Zhang worked on the seller's product uploading, viewing, updating, order viewing, and updating the page. Yizhan Xue worked on the seller's sales report, administrators' main page, current order page, and information updating page. We insisted on meeting weekly to discuss the project's progress, and when teammates needed help, everyone actively helped to solve issues.