Project Report

Mark Kim:

I worked on 3 main implementations.

1) The Menu() function
2) The UpdateOrder() function
3) The BrowsingOrderHistory() function

1) The Menu() function has two main functionalities being:
    a) Allowing users to see and search the menu
    b) Allowing Managers to Add/Delete/Update Items

I created the functions ItemMenu(), EnterItemName(), and EnterItemType() to allow users to see and search(by item name and type) the menu. In hindsight, I could have squeezed these 3 functions into just one, thus cleaning up the implementation. I worked with this in mind in future functions, however, this implementation was my first one for this phase, so I just left it in its state. EnterItemName and EnterItemType each have one SQL query to find and print all items with the inputted name or type.

I couldn't figure out a good way to check whether a user was a customer, manager, or employee, so I just asked the user to input login and password again to verify if they are a manager using an SQL query. Adding and Deleting items was simple, asking for an input for each attribute to add an item using an INSERT INTO query and asking for the primary key itemName to delete an item using DELETE query. To update an item, I asked the user for the itemName and then showed a table listing all the attributes and asking which one the user would like to change. Each attribute update utilizes the UPDATE query with the primary key itemName. This section uses functions ManagerLogin() and UpdateItem().

2) The UpdateOrder() function was relatively simple as well.

I first asked the user to input the OrderID of the item they would like to update, regardless of which type of user they were, this was an essential step. From there, a table listing the attributes of an order is shown and then the user chooses the attribute they would like to update. I didn't know if customers should be allowed to change

attributes like timeStampReceived or even OrderID, however I just let every user change any attribute since the documentation did not specify.

For changing the ‘paid’ attribute, users are prompted to login again, thus being able to verify that they are not a customer, and thus a manager or employee. Once verified, the ‘paid’ attribute immediately changes to true and exits. Of course, every update involves using an UPDATE query.

3) BrowsingOrderHistory() has just two functionalities. One being allowing users to see their last 5 orders. The other being allowing Managers and Employees to see all unpaid orders within the last 24 hours. Since Managers and Employees are instructed to be able to see all unpaid orders within 24 hours, I made viewing the last 5 recent orders to be a Customer-exclusive functionality. Utilizing LIMIT 5 in a SELECT SQL query, I was able to output the last 5 orders a Customer has made given their login.

As for the second functionality, I had a hard time figuring out how to implement the 24 hours aspect. I used the internet to figure out this SQL query:

SELECT * FROM ORDERS WHERE paid = ‘false’ and timeStampReceived > now() - interval ‘24 hours’

The problem here is that I didn’t know whether to use our current time in the real world or use something like the time of the last order. I ended up using the current time in the world, but in that case, none of the test data would output anything since the timestamp for those orders are in 2016. In this case, I would have to input my own data and thus test it.

Gonzalo Ruiz: I worked on the user updating their profile and placing an order implementation. First of all, for updating the profile, since the user is updating sensitive data like password etc. I require the user to enter login and password again for safety. Then, the user is presented with a menu that allows them to update their user info like login, password, phone number etc. Additionally, if the user is a manager he/she can update the type of user, for any user. The manager will be prompted to enter the login of the user he/she wants to update to either: manager, employee or customer. Finally, the table is presented to the manager showing the updated user type for confirmation. Some problems I had during this implementation was that I was trying to write query like this: ***String query = String.format("SELECT * FROM USERS WHERE type = Manager”);*** To check if user is a manager, however this keep causing an error and so what I found out is that I had to format like this:

String type="Manager";
String query = String.format("SELECT * FROM USERS WHERE type = '%s'", type);
So these were some of the problems/findings I found when implementing this part.

For placing an order implementation, I basically do an insertion into the Orders table and set all of the values to default. So for example, set the value 'paid' to false, total to 0, and timestamp to now. I also modified the function ***PlaceOrder(Cafe esql)*** to include the user login and insert it into the orders table, like this: ***PlaceOrder(Cafe esql, String login)***. The tricky part for this was the orderid, because it has to be unique and not null. So I created a sequence starting at the last value of order id in the orders table. I also created a trigger to update orderid before an insert (similar to lab 8). For this I created a new file ***triggers.sql*** at ***project/sql/src/triggers.sql*** and modified ***project/sql/scripts/create_db.sh*** to include this new file. Then the user is asked which item from the menu they wish to order and I update the total for the new order insertion to the price of the item from the menu. Finally, the program shows the orders this user has active and shows the new order placed. Some of the problems I found in this implementation was that I needed the orderid to update the total price and I wasn't sure how to get it since the only place orderid is updated was in the sequence trigger where I access it using nextval(). Reading the documentation from lab8 I found out about the function currval(). This function allowed me to get the orderid of the order that was just created and to update the total with the item price from the menu.