Database Design Report

Business Rules:

Staff and Orders – One-to-many Relationship:

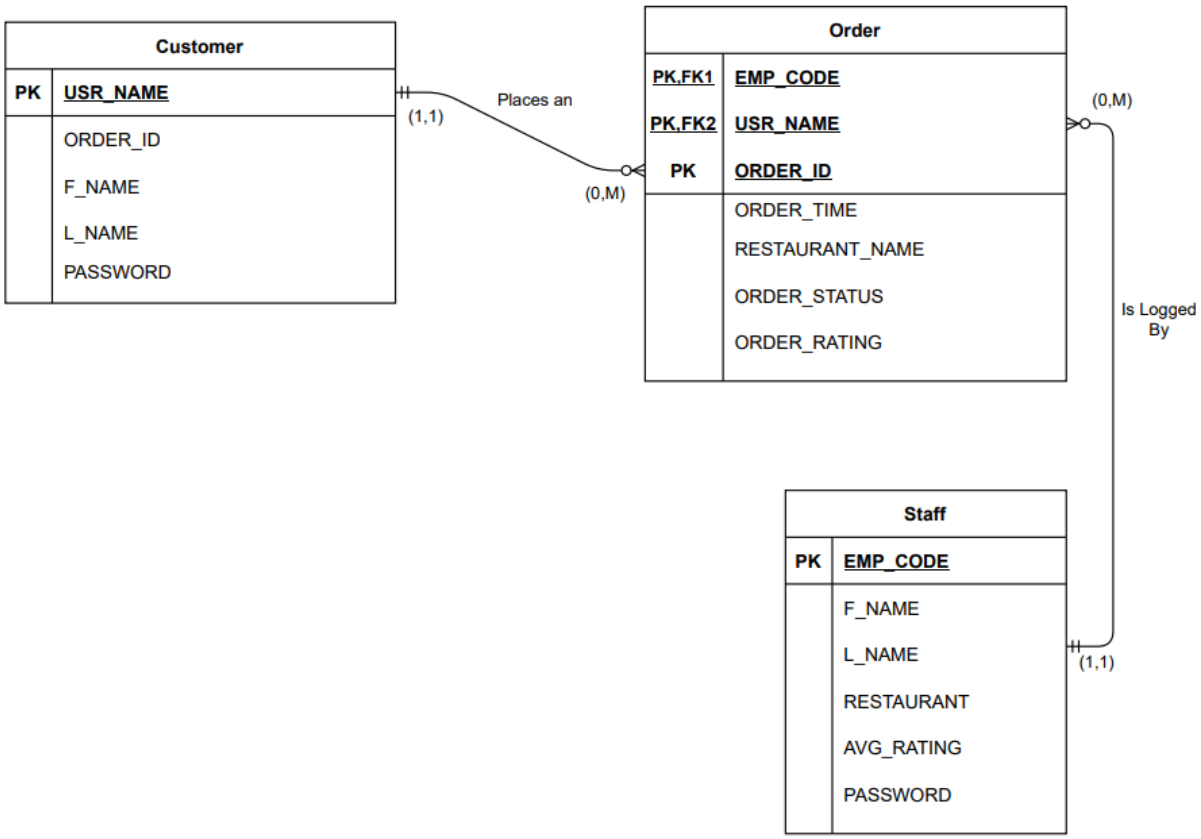
Each Staff member is able to log many/multiple orders, but each Order must be logged by one and only one staff member. Staff (1:M) : Orders (1:1) => 1:M

Customer and Orders – One-to-many Relationship:

Each Customer is able to place many orders, but each order must be placed by one and only one customer. Customer (1:M) : Orders (1:1) => 1:M

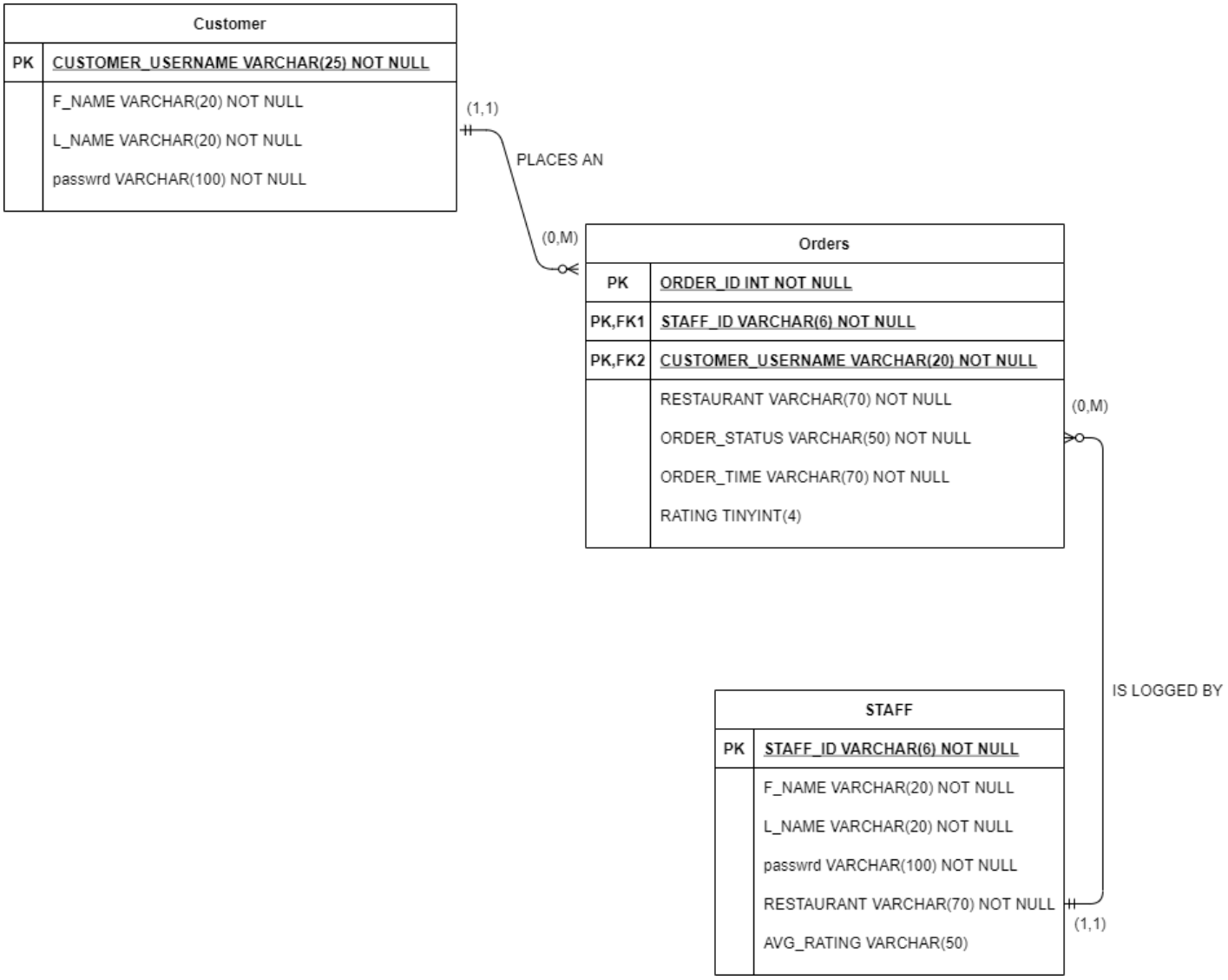
Entity Relationship Diagrams (ERDs):

Initial ERD



Secondary ERD:

There were some changes made to certain attribute characteristics from the initial ERD.



Issues such as M: N relationships, NULL values, multivalued attributes:

One customer would be able to place many orders through many staff members and one staff member is able to log many orders for many customers, therefore creating many-to-many relationships (M: N). Therefore, a bridge entity, Orders, was created to maintain one-to-many relationships throughout the database.

Customer table:

1. Used to store details of each customer.
2. Primary key Is the customers username as each customer needs to have a unique username which they can be identified as.

Staff table:

1. Staff table is used to store all the details of staff members.
2. Primary key is the staff's ID as each staff member is assigned a unique ID which can be used to identify a particular member of staff.

Orders table:

1. The orders table is used to log and keep track of all orders.
2. The attribute RATING (Boolean) stores the rating of that particular order as a "1" if the order is rated thumbs up, "0" if order is rated thumbs down or "null" if no rating yet. Although null values should be avoided wherever possible, in this case we need to have the possibility of a null value as an order is not required to have a rating. It's also important to be able to have a null value as when calculating the staff members average rating, orders that are null will not be included.
3. For the Orders table we have 3 primary keys, the staff member who logged the orders' id, the customer who placed the orders' username and lastly the order id. This is because we have used the orders table as a bridge entity and thus it must maintain its connection to the parent tables. The staff Id and customer username are also foreign keys in the table so that we have access to all the relevant information of the staff and customer involved with the order.

Implementation of database:

```
mysql> CREATE TABLE STAFF
    -> (
    ->     STAFF_ID VARCHAR(6),
    ->     F_NAME VARCHAR(20) NOT NULL,
    ->     L_NAME VARCHAR(20) NOT NULL,
    ->     RESTAURANT VARCHAR(70) NOT NULL,
    ->     passwrd VARCHAR(150) NOT NULL,
    ->     AVG_RATING VARCHAR(50),
    ->
    ->     PRIMARY KEY(STAFF_ID)
    -> );
Query OK, 0 rows affected (0.05 sec)

mysql> CREATE TABLE CUSTOMER
    -> (
    ->    CUSTOMER_USERNAME VARCHAR(25),
    ->    passwrd VARCHAR(150) NOT NULL,
    ->    F_NAME VARCHAR(20) NOT NULL,
    ->    L_NAME VARCHAR(20) NOT NULL,
    ->
    ->    PRIMARY KEY(CUSTOMER_USERNAME)
    -> );
Query OK, 0 rows affected (0.06 sec)

mysql> CREATE TABLE ORDERS
    -> (
    ->     STAFF_ID VARCHAR(6),
    ->     CUSTOMER_USERNAME VARCHAR(20),
    ->     ORDER_ID INT,
    ->     RESTAURANT VARCHAR(50) NOT NULL,
    ->     ORDER_STATUS VARCHAR(30) NOT NULL,
    ->     RATING TINYINT,
    ->     ORDER_TIME VARCHAR(70) NOT NULL,
    ->
    ->     FOREIGN KEY(STAFF_ID) REFERENCES STAFF(STAFF_ID),
    ->     FOREIGN KEY(CUSTOMER_USERNAME) REFERENCES
CUSTOMER(CUSTOMER_USERNAME),
    ->     PRIMARY KEY(ORDER_ID)
    -> );
Query OK, 0 rows affected (0.10 sec)
```

```
mysql> DESC STAFF;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| STAFF_ID   | varchar(6)   | NO   | PRI |         |       |
| F_NAME     | varchar(20)  | NO   |     | NULL    |       |
| L_NAME     | varchar(20)  | NO   |     | NULL    |       |
| RESTAURANT | varchar(70)  | NO   |     | NULL    |       |
| passwrd    | varchar(150) | NO   |     | NULL    |       |
| AVG_RATING | varchar(50)  | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
6 rows in set (0.00 sec)

mysql> DESC CUSTOMER;
+-------------------+--------------+------+-----+---------+-------+
| Field             | Type         | Null | Key | Default | Extra |
+-------------------+--------------+------+-----+---------+-------+
| CUSTOMER_USERNAME | varchar(25)  | NO   | PRI |         |       |
| passwrd           | varchar(150) | NO   |     | NULL    |       |
| F_NAME            | varchar(20)  | NO   |     | NULL    |       |
| L_NAME            | varchar(20)  | NO   |     | NULL    |       |
+-------------------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> DESC ORDERS;
+-------------------+-------------+------+-----+---------+-------+
| Field             | Type        | Null | Key | Default | Extra |
+-------------------+-------------+------+-----+---------+-------+
| STAFF_ID          | varchar(6)  | YES  | MUL | NULL    |       |
| CUSTOMER_USERNAME | varchar(20) | YES  | MUL | NULL    |       |
| ORDER_ID          | int(11)     | NO   | PRI | 0       |       |
| RESTAURANT        | varchar(50) | NO   |     | NULL    |       |
| ORDER_STATUS      | varchar(30) | NO   |     | NULL    |       |
| RATING            | tinyint(4)  | YES  |     | NULL    |       |
| ORDER_TIME        | varchar(70) | NO   |     | NULL    |       |
+-------------------+-------------+------+-----+---------+-------+
7 rows in set (0.01 sec)
```

SQL statements used:

Since all SQL statements used in the project other than the table implementation, were done in php I have shown below all the statements used in the php scripts in order to interact with the database.

Staff Signup:

“SELECT * FROM $userType WHERE STAFF_ID='$STAFF_ID' AND passwrd='$passwrd'”;

Staff Login:

SELECT * FROM $userType WHERE STAFF_ID='$STAFF_ID' AND passwrd='$passwrd'";

Checking for unique staff ID:

"SELECT STAFF_ID FROM $userType";

Checking unique order ID:

"SELECT ORDER_ID FROM $userType";

Customer login:

"SELECT * FROM $userType WHERE CUSTOMER_USERNAME='$CUSTOMER_USERNAME' AND passwrd='$passwrd'";

Checking that customer username is unique:

"SELECT CUSTOMER_USERNAME FROM $userType";

Customer signUp:

"INSERT INTO CUSTOMER VALUES('$CUSTOMER_USERNAME', '$passwrd', '$F_NAME', '$L_NAME')";

Getting the average rating of the orders of a specific staff member:

"SELECT AVG(RATING) FROM ORDERS WHERE STAFF_ID='$STAFF_ID'";

Placing order:

"INSERT INTO ORDERS VALUES('$STAFF_ID', '$CUSTOMER_USERNAME', '$ORDER_ID', '$RESTAURANT','$ORDER_STATUS',NULL,'$ORDER_TIME')";

Requesting order information of a customer:

"SELECT * FROM ORDERS WHERE CUSTOMER_USERNAME='$CUSTOMER_USERNAME'";

Rating order with a thumbs up:

"UPDATE ORDERS SET RATING=1 WHERE ORDER_ID='$ORDER_ID'";

Rating order with thumbs down:

"UPDATE ORDERS SET RATING=0 WHERE ORDER_ID='$ORDER_ID'";

Setting the average rating of a staff member:

"UPDATE STAFF SET AVG_RATING='$AVG_RATING' where STAFF_ID='$STAFF_ID'";

Getting information about a specific staff member:

"SELECT F_NAME,L_NAME FROM STAFF WHERE STAFF_ID='$STAFF_ID'";

Getting orders associated with a specific staff member:

"SELECT * FROM ORDERS WHERE STAFF_ID='$STAFF_ID'";

Changing the order status:

"UPDATE ORDERS SET ORDER_STATUS='$STATUS' WHERE ORDER_ID='$ORDER_ID'";

Data Validation:

Before a query is made to the database, the data is validated. The validation occurs in the android code. We have used the database to check that the order IDs, staff IDs and customer usernames are unique by making a query to the database to get all current entries In each respective filed and then checking that the ID/username we want to register or use are not already present. If they are not found then we proceed with the task at hand, however if a value is found that matches, an error message is displayed alerting the user of the issue.

Procedures and process:

When opening the app, the first thing you are presented with is the option to be a customer or staff member. Depending on which option you choose you will be directed to the next relevant screen. However, regardless of which choice you make you will be presented with a login screen, if you do not have an account you can also choose to sign up.

When signing up as a customer you will be required to enter your first name, last name, username and password. If your username is unique-which is achieved by checking the entered username against all current usernames in the database- you will be successfully signed up and may proceed to login. The same is true for the staff sign up, however you must enter the restaurant for which you work, and rather than entering a username a staff ID will be generated for you.

After logging in as a customer you will be presented with a view containing all your orders. The orders will be categorized by their status, either pending, ready or collected. Upon pressing the order, you will be sent to a new activity with all the relevant order details such as order time, order ID, staff member who logged the order and their average rating. This is done by sending a request to the database when the order is clicked requesting the information about that specific order based on the order ID. The customer may then rate the order by clicking on either the thumbs up/down button. This will again trigger a call to the database logging the rating in the relevant entry of the orders table.

After logging in as a staff member you will be presented with the staffs main screen which is a scroll view containing all the orders the have logged for various customers and a button which will take them to a new activity which they can log new orders from. If the staff member clicks on one of the orders they will be able to change the status of the order, if they do so, this will send a query to the database changing the order status which will in turn change the location of the order to the relevant view in the customers main page. If the staff member clicks on the add order button they will be taken to the create orders page where they will be required to enter the username of the customer for whom they are logging the order. Once the place order button is clicked a query is sent to the database logging the new order in the order table with all the relevant information grabbed form the create orders page.