DBF/ MC Project : Restaurant App - Order Track

17/06/2021

—

Done By:
Zagesh Parshotam (1845347)
Lerusha Munien (2305631)

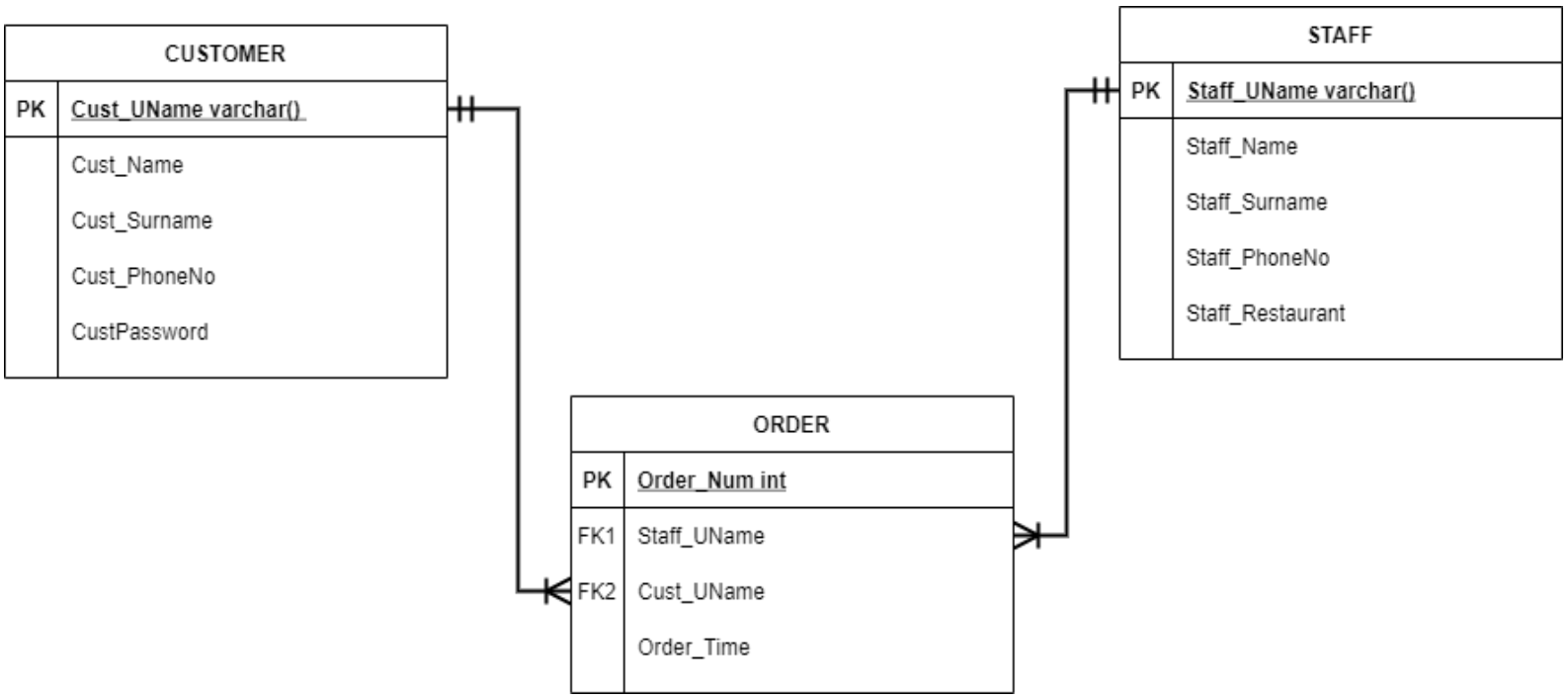
Overview

Our restaurant application can track your orders made to the various restaurants. Both the customer and staff have their credentials for this application to log in and view different parts of the application.

Specifications

The application should be able to login to staff or customers or register a new staff or client. They will be directed to their respective screens. The customer will be able to view all their orders, quite similar to order history. They will also be able to rate their orders using a thumbs up or down feature. On the other hand, the staff will be able to add a new order as well as update the status of ongoing orders so the customer is aware of when their order is ready. The staff will also be able to see their average rating from all their orders.

Initial ERD



Database

In our database, there are 3 tables, namely CUSTOMER, STAFF, and tblORDER. We have made sure to do data normalization to make sure there are little or no data redundancy and anomalies created.

★ Customer Table

This table is used to store all the details of each customer. It contains attributes such as Cust_UName for their username, Cust_FName for their name, Cust_PhoneNo for their contact number, and CustPassword for their password. The primary key is Cust_UName as no two customers are allowed to have the same username thus making it unique.

★ Staff Table

This table is used to store all the details of each staff member. It contains attributes such as Staff_UName for their username, Staff_FName for their name, Staff_PhoneNo for their contact number, and Staff_Password for their password. The primary key is Staff_UName as no two staff members are allowed to have the same username thus making it unique.

★ tblORDER Table

This table will be used to record all the orders made. It contains attributes such as Order_Num for the order number that is auto-incremented, Order_Time for the time and dates the order was created, Order_Status which stores the status of the order(Pending, Ready or Collected), Order_Rating which stores the rating of the order made by the customer, Order_Rest for the restaurant the order is from, Staff_UName for the staff who is in charge of the order, and lastly Cust_UName for the customer who placed the order. The primary key is Order_Num as it will be used to uniquely identify each record. The foreign keys are Staff_UName from the Staff table and Cust_UName from the Customer table.

Business Rules

One customer can place many orders but each order can only be placed by one customer at a time.

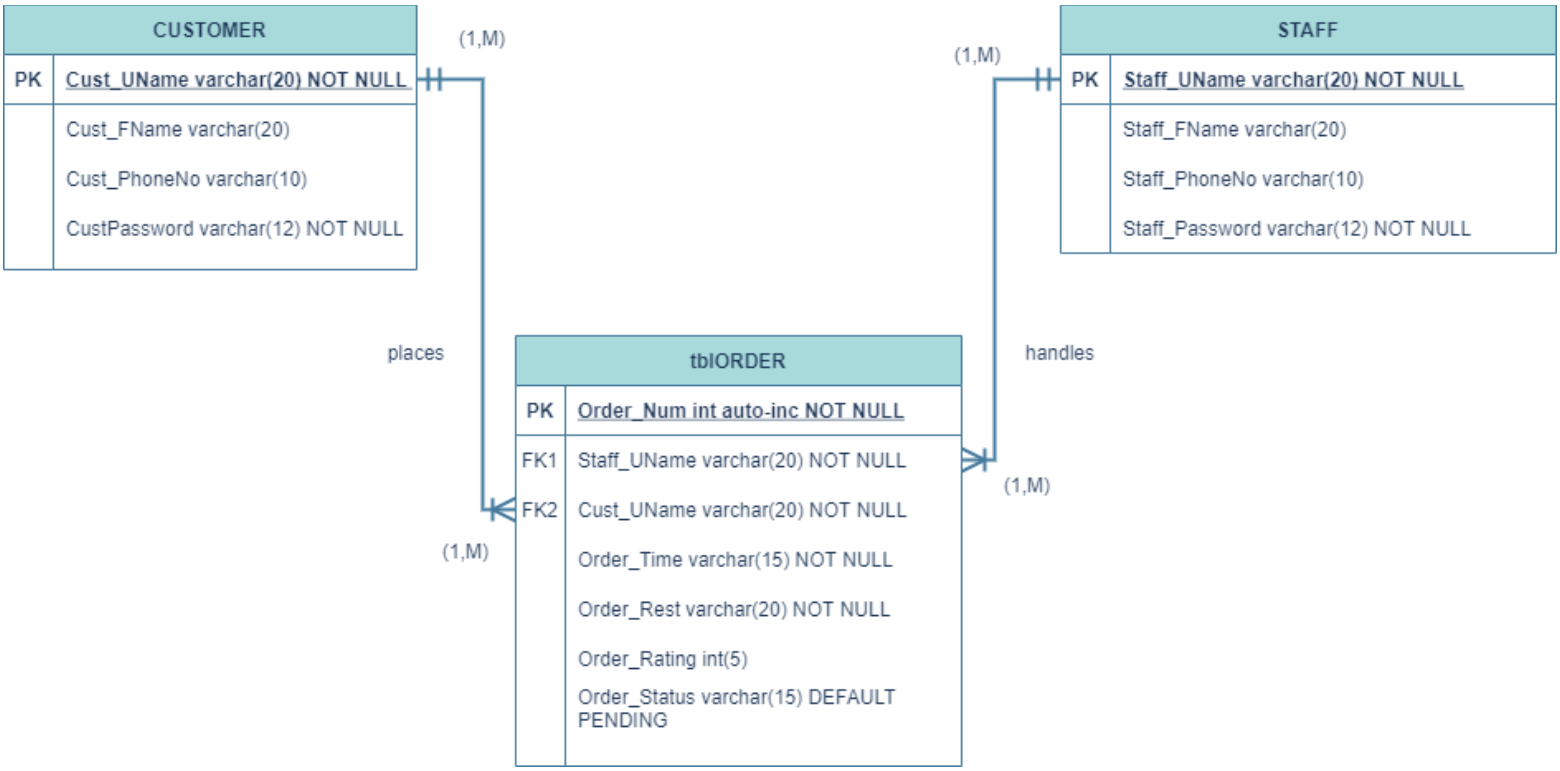
CUSTOMER (1:M) : tblORDER (1:1) => (1:M)

One staff member can handle many orders, but each order can only be handled by one staff member.

STAFF (1:M) : tblORDER (1:1) => (1:M)

Final ERD

We choose not to include the cardinalities in the form of a number instead of M in the ERD due to the fact that any number of orders should be placed in a day.However, there cannot be too much where the staff members are being overworked.



Implementation

★ Tables

```
mysql> CREATE TABLE CUSTOMER (Cust_UName varchar(20) NOT NULL PRIMARY KEY,
Cust_FName varchar(20) , Cust_PhoneNo varchar(10),CustPassword varchar(12)
NOT NULL);

Query OK, 0 rows affected (0.11 sec)


mysql> CREATE TABLE STAFF (Staff_UName varchar(20) NOT NULL PRIMARY
KEY,Staff_FName varchar(20),Staff_PhoneNo varchar(10),Staff_Password
varchar(12) NOT NULL);

Query OK, 0 rows affected (0.04 sec)


mysql> CREATE TABLE tblORDER(Order_Num int NOT NULL AUTO_INCREMENT PRIMARY
KEY,Staff_UName varchar(20) NOT NULL,Cust_UName varchar(20) NOT
NULL,Order_Time varchar(15) NOT NULL,Order_Rest varchar(20) NOT
NULL,Order_Status DEFAULT PENDING,Order_Rating int);

Query OK, 0 rows affected (0.28 sec)


mysql> ALTER TABLE tblORDER ADD FOREIGN KEY (Staff_UName) REFERENCES
STAFF(Staff_UName);

Query OK, 0 rows affected (0.13 sec)

Records: 0  Duplicates: 0  Warnings: 0


mysql> ALTER TABLE tblORDER ADD FOREIGN KEY (Cust_UName) REFERENCES
CUSTOMER(Cust_UName);

Query OK, 0 rows affected (0.28 sec)

Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> INSERT INTO CUSTOMER
VALUES('lerushisonfire','Lerusha','0678521230','test123');
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO STAFF VALUES('ziggy','Zagesh','0823652610','test456');
Query OK, 1 row affected (0.01 sec)

mysql> desc STAFF;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| Staff_UName   | varchar(20) | NO   | PRI | NULL    |       |
| Staff_FName   | varchar(20) | YES  |     | NULL    |       |
| Staff_PhoneNo | varchar(10) | YES  |     | NULL    |       |
| Staff_Password| varchar(12) | NO   |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)

mysql> DESC CUSTOMER;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| Cust_UName  | varchar(20) | NO   | PRI | NULL    |       |
| Cust_FName  | varchar(20) | YES  |     | NULL    |       |
| Cust_PhoneNo| varchar(10) | YES  |     | NULL    |       |
| CustPassword| varchar(12) | NO   |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

```
mysql> DESC tblORDER;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| Order_Num    | int         | NO   | PRI | NULL    | auto_increment |
| Staff_UName  | varchar(20) | NO   | MUL | NULL    |                |
| Cust_UName   | varchar(20) | NO   | MUL | NULL    |                |
| Order_Time   | varchar(15) | NO   |     | NULL    |                |
| Order_Rest   | varchar(20) | NO   |     | NULL    |                |
| Order_Status | varchar(15) | YES  |     | PENDING |                |
| Order_Rating | int         | YES  |     | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
7 rows in set (0.00 sec)
```

```
mysql>DELIMITER //
mysql>
mysql>
mysql> CREATE FUNCTION AVG_RATING(ORD_RATING DOUBLE(4,4))
    -> RETURNS INT
    -> BEGIN
    -> DECALRE ANS INT(2)
    -> SET ANS = 0;
    -> SET ANS = ROUND(ORD_RATION *100)
    -> RETURN ANS
    -> END; //
```

★ Other SQL statements used in PHP and Functions

1. $query = mysqli_prepare($link, "SELECT * FROM tblORDER WHERE Staff_UName=?)

2. $sql = "INSERT INTO ORDER(Order_Time,Order_Rest,Cust_UName,Staff_UName) VALUES (?, ?, ?, ?)";

3. $sql = "SELECT * FROM tblORDER WHERE Cust_UName= ?";

4. $sql = "INSERT INTO CUSTOMER (Cust_UName,Cust_FName,Cust_PhoneNo,Custpassword) VALUES (?, ?, ?, ?)";

5. $sql = "UPDATE tblORDER SET Order_Status =? WHERE Order_Num = ?";

6. $sql = "SELECT AVG_RATING(AVG(Order_Rating)) as 'Your average rating' from tblORDER where Staff_UName ='$username'";

7. $sql = "INSERT INTO STAFF(Staff_UName,Staff_FName,Staff_Password,Staff_PhoneNo) VALUES (?, ?, ?, ?)";

8. $query = mysqli_prepare($link, "SELECT Order_Num, Order_Time, Order_Rest, Order_Status, Order_Rating FROM tblORDER WHERE Cust_UName = ? ORDER BY Order_Num DESC");

9. $sql = "SELECT * FROM tblORDER WHERE Order_Num= ?";

10. $query = mysqli_prepare($link, "SELECT * from CUSTOMER where Cust_UName=? and CustPassword=?");

11. $query = mysqli_prepare($link, "SELECT * from STAFF where Staff_UName=? and Staff_Password=?");

12. $sql = "UPDATE tblORDER SET Order_Rating = ? WHERE Order_Num = ?";

13. mysql>DELIMITER //
    mysql>
    mysql>
    mysql> CREATE FUNCTION AVG_RATING(ORD_RATING DOUBLE(4,4))
        -> RETURNS INT
        -> BEGIN
        -> DECALRE ANS INT(2)
        -> SET ANS = 0;
        -> SET ANS = ROUND(ORD_RATION *100)
        -> RETURN ANS
        -> END; //

Data Validation

Any data entered on the application by the staff member or customer is validated before being entered into the database. This validation occurs in the PHP script. We have checked whether the data entered is empty, already exists for example a username that is unique, or if the data contains certain special characters which should not be required. Also specifically for passwords, we have validated the length to make sure the password is strong. If there is an error in the data the user has entered, the relevant error messages are shown to alert the user about their input. Below is an example of the data validation that occurred before inserting a new customer into the table CUSTOMER.

```
if(empty(trim($username))){
    array_push($errors, "Username is required.");
    echo "Username is required.";
```

```
} elseif(!preg_match('/^[a-zA-Z0-9_]+$/', trim($username))){
        array_push($errors, "Username can only contain letters, numbers, and underscores.");
        echo "Username can only contain letters, numbers, and underscores.";
}elseif(empty(trim($fname))){
        array_push($errors, "Please enter your name.");
        echo "Please enter your name.";
}elseif(empty(trim($phonenum))){
        array_push($errors, "Please enter a phone number.");
        echo "Please enter a phone number.";
}elseif(strlen($phonenum)<10){
        array_push($errors, "Incorrect phone number.");
        echo "Incorrect phone number.";
}elseif(empty(trim($password))){
        array_push($errors, "Please enter a password.");
        echo "Please enter a password.";
} elseif(strlen(trim($password)) < 6){
        array_push($errors, "Password must have atleast 6 characters.");
        echo "Password must have atleast 6 characters.";
} else{
        // Prepare a select statement
        $sql = "SELECT * FROM CUSTOMER WHERE Cust_uName= ?";

        if($stmt = mysqli_prepare($link, $sql)){
                // Bind variables to the prepared statement as parameters
                mysqli_stmt_bind_param($stmt, "s", $username);

                if(mysqli_stmt_execute($stmt)){
                        /* store result */
                        mysqli_stmt_store_result($stmt);
```

```
                if(mysqli_stmt_num_rows($stmt) == 1){
                    array_push($errors, "This username is already used.");
                }

            }

            // Close statement
            mysqli_stmt_close($stmt);
        }
}
```

Apart from data validation, we have also made sure to bind the parameters being sent into the SQL statements. This way we will prevent any SQL injections. An example is shown below.

```
$sql = "INSERT INTO CUSTOMER (Cust_UName,Cust_FName,Cust_PhoneNo,Custpassword)
VALUES (?, ?, ?, ?)";
    if($stmt = mysqli_prepare($link, $sql))
    {
        // Bind variables to the prepared statement as parameters

        mysqli_stmt_bind_param($stmt, "ssss", $username, $fname, $phonenum, $password);

        // Attempt to execute the prepared statement
        if(mysqli_stmt_execute($stmt))
        {
            echo "Registration successful";
        }
        mysqli_stmt_close($stmt);
    }
```

About our final product: Procedures and Processes

Upon opening the application, you will first see a splash screen with our application and logo name, Order Track. Next, you will see the login screen where the customer or staff member can enter their credentials. After the credentials such as your username, and password have been validated in the PHP file, this then triggers a database search action where a check is done to see whether the data is correct in either the staff or customer table. If so it will allow the user to the respective screens.

There is also a button to sign up if you do not have an account. Here you will be asked to enter details such as a username, name, phone number, and password. This then triggers a database insertion into the table customer or staff if all data entered is correct when it is validated. Remember that no two users may have the same username so you will not proceed if that's the case. From here you will be directed to your respective screens.

In addition to this, there is a button to click if you forgot your password. This means the user can change their passwords and validation will also occur in the PHP file before a database insertion into the respective table is made.

On the customer screen, you will be able to view all your orders, quite similar to order history. This is done by displaying all the records from the table customers for the customer username entered. You will have a chance to rate your respective order, by clicking a thumbs up or down, which then triggers a database action of inserting your rating into the tblORDER table. The orders will appear in white if they have been collected and given a rating.

On the staff screen, the staff member will be able to create a new order, by entering relevant details such as the customer username, their staff username, and the restaurant using a dialog box. This will trigger a database insertion into tblORDER after validating the data entered. And in addition to this, the staff member will be able to view all their orders from the most recent order. This means all records in descending order will be shown from tblORDER. Each staff can also update the order status by entering the order number and its status into a dialog box they will be prompted with. This will cause an update to occur in the tblORDER table for the Order status. The orders will appear in white if they have been collected and given a rating. Lastly, the staff member will also be able to see their average rating as a percentage and this is caused by a function created in SQL called AVG_RATING. This function is called in the PHP file and displays each staff member's rating.