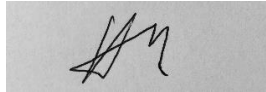


Documentation Report

January 6, 2019

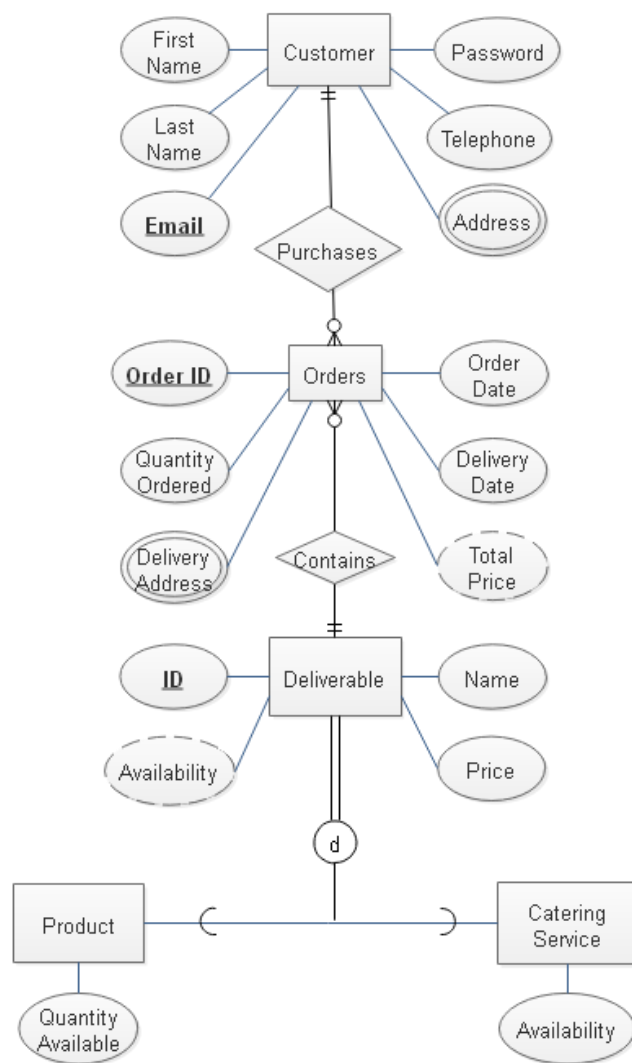
I confirm that all of the content in this report is my own work

A handwritten signature in black ink, appearing to be 'AM', is centered on a light gray rectangular background.

1 Developing and Testing Environment

- The software I used to draw my EER diagram was **Edraw Max 9.3**
- The software I used to draw my database schema diagram was a website called draw.io
- I used the MySQL server included in **ApacheFriends XAMPP Version 7.2.12** for this assignment.
 - Download link:
<https://sourceforge.net/projects/xampp/files/XAMPP%20Windows/7.2.12/>
- I used **MySQL Workbench 8.0** to manually code my database and test it.
- I used the **Eclipse Photon IDE for PHP Developers** to develop all of my website.
- I used the Apache web server and the PHP module included in **ApacheFriends XAMPP Version 7.2.12** to test my website (download link in bullet 3).
- The web browser I used to test my website was **Google Chrome Version 71.0.3578.98**
- The operating system I used during all of my development and testing processes was **Windows 10 Home**.

2 EER Data Model and Diagram



In this EER diagram, there are 5 key entity types:

- The **Customer** with all of his/her details/attributes including a multi-valued attribute for the customer's address details. Each customer is identified using the email attribute.
- An **Order** (had to be named Orders because MySQL has an Order keyword) with its details/attributes including a multi-valued attribute for the order's delivery address. Each order is identified using the Order ID attribute.
- A **Deliverable** supertype which can either be (be linked to) only a **Product** subtype or a **Catering Service** subtype. A deliverable's details/attributes are also the **Product**'s and **Catering Service**'s details/attributes. Every **Deliverable** is identified using the ID attribute.
- A **Product** subtype which has one of its own attributes: Quantity Available.
- A **Catering Service** subtype which has one of its own attributes: Availability.

The relationships that link together these entity types are:

- A **binary** relationship between **Customer** and **Orders**; Every customer can purchase an order.
 - Cardinality constraints: **one-to-many**; one customer can purchase many orders.
 - Participation constraints: **partial participation** for **Customer** and **total participation** for **Orders**; A customer may or may not wish to purchase (make) an order, but an order will only exist when a customer decides to make it and will be tied to that one customer.
- A **binary** relationship between **Orders** and **Deliverable**; Every order contains one kind of deliverable.
 - Cardinality constraints: **one-to-many**; one kind of deliverable can be part of many different orders.
 - Participation constraints: **total participation** for **Orders** and **partial participation** for **Deliverable**; An order must contain one kind of deliverable, but a deliverable can exist without being part of an order.
- A special **ternary** relationship between **Deliverable**, **Product** and **Catering Service**; A deliverable can be and only be either a product or catering service.
 - Cardinality constraints: **disjoint**; either **Product** or **Catering Service**.
 - Participation constraints: **total participation**; Every deliverable must be a product or catering service.

3 Conceptual/Logical Relational Database Schema

Customer(FirstName:String, LastName:String, Email:String, Password:String, Telephone:String)

Address(Email:String, Street:String, HouseNumber:String, Area:String, ZIPCode:String)

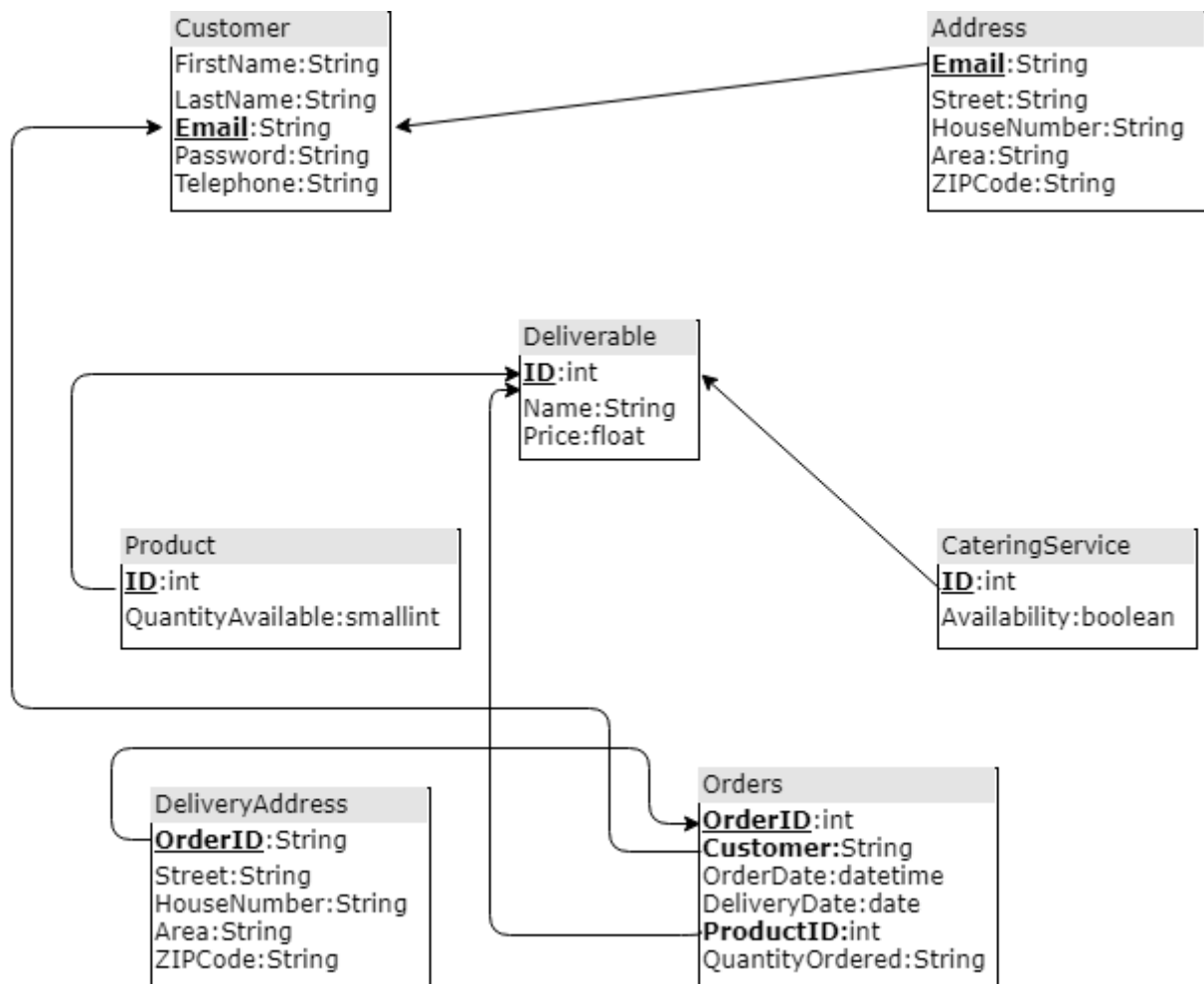
Deliverable(ID:int, Name:String, Price:float)

Product(ID:int, QuantityAvailable:smallint)

CateringService(ID:int, Availability:boolean)

Orders(OrderID:int, Customer:String, OrderDate:datetime, DeliveryDate:date, ProductID:int, QuantityOrdered:String)

DeliveryAddress(OrderID:int, Street:String, HouseNumber:String, Area:String, ZIPCode:String)



4 MySQL Code

MySQL Code Used for Creating the Database:

- Create statement to make database exist, then creating each table(relation) with its columns(attributes) and corresponding datatypes and parameters. The tables were created in order of relationship; so that if a table has a foreign key that references another table, that other table being referenced would have been created before it.
 - In Customer table I chose varchar to be datatype for Telephone column because MySQL does not include any zeros before number values.
 - In Address and DeliveryAddress tables I chose varchar to be datatype for HouseNumber because some house numbers have letters in them ex: 66J and additionally, I allowed ZIPCode column to take null values because in my country, zip codes aren't really used to identify addresses so customers don't have to enter a zip code if they wish.
 - In Orders table I set OrderID, which is the primary key, to auto increment so that when a new order is placed in the table, the order would have a unique id automatically.
 - Also, in Orders table, I set datatype of QuantityPurchased to varchar because if catering service is purchased, value needs to be in form of "x people".

MySQL Code Used for Populating the Database:

- In this website, customers have the ability to choose what they would like to purchase from two sets of Deliverable things: a physical product or a catering service. Therefore, populating the database included:

- Inserting values in the Deliverable, Product and CateringService tables.

Embedded MySQL Code in my PHP Code:

Code in *account.php*

- `SELECT OrderID FROM Orders WHERE Customer='$email'`
 - Used to check whether or not a certain customer has made any orders at all.
- `SELECT * FROM Orders INNER JOIN DeliveryAddress ON Orders.OrderID = DeliveryAddress.OrderID INNER JOIN Deliverable ON Orders.ProductID = Deliverable.ID WHERE Orders.Customer = '$email' ORDER BY Orders.OrderDate DESC`
 - Used to retrieve all orders and their corresponding product or service and their delivery address for a certain customer and then order each order by when it was made.

Code in *cateringorder.php*

- `SELECT Availability FROM CateringService WHERE ID = $product`
 - Used to check whether chosen catering service is currently available or not.
- `INSERT INTO Orders(Customer, OrderDate, DeliveryDate, ProductID, QuantityOrdered) VALUES ('$user', '$date', '$deliveryDate', $product, '$quantityInsert')`
 - Used to insert an order made by a certain customer, that involved purchasing a catering service, into the Orders table.
- `SELECT OrderID FROM Orders WHERE Customer = '$user' AND OrderDate = '$date' AND DeliveryDate = '$deliveryDate' AND ProductID = $product AND QuantityOrdered = '$quantityInsert'`
 - Used to retrieve the order id that was generated when a new order is inserted in the Orders table, so that it can be used as a foreign key later in the DeliveryAddress table. (This purpose could have been achieved without needing SQL statement by setting the OrderID column in DeliveryAddress to auto increment as well every time an address is inserted directly after an order has been inserted, but I used this statement just to be 100% sure when matching the delivery address with the rest of the order).
- `INSERT INTO DeliveryAddress(OrderID, Street, HouseNumber, Area, ZIPCode) VALUES ($orderID, '$street', '$house', '$area', '$zipCode')`
 - Used to insert a delivery address in the DeliveryAddress table, which corresponds with a certain order in the Order table.
- `SELECT Name, Price FROM Deliverable WHERE ID = $product`
 - Used to retrieve the name and price of a chosen catering service so that they could be later stored in a session variable containing all details of the order made which will be displayed in a receipt.

Code in *productorder.php*

- `SELECT QuantityAvailable FROM Product WHERE ID = $product`
 - Used to check whether chosen product has enough stock currently for the customer's demanded quantity, and store the current product's stock in a variable for future use.
- `INSERT INTO Orders(Customer, OrderDate, DeliveryDate, ProductID, QuantityOrdered) VALUES ('$user', '$date', '$deliveryDate', $product, '$quantity')`

- Same purpose as bullet #2 in *cateringorder.php* above, except it is inserting an order that involved purchasing a product.
- `SELECT OrderID FROM Orders WHERE Customer = '$user' AND OrderDate = '$date' AND DeliveryDate = '$deliveryDate' AND ProductID = $product AND QuantityOrdered = '$quantity'`
 - Same purpose as bullet #3 in *cateringorder.php* above.
- `INSERT INTO DeliveryAddress(OrderID, Street, HouseNumber, Area, ZIPCode) VALUES ($orderID, '$street', '$house', '$area', '$zipCode')`
 - Same purpose as bullet #4 in *cateringorder.php* above.
- `UPDATE Product SET QuantityAvailable = $newQuantity WHERE ID = $product`
 - Used to update the QuantityAvailable column for a chosen product with its new quantity available after a purchase has been made for that product.
- `SELECT Name, Price FROM Deliverable WHERE ID = $product`
 - Same purpose as bullet #5 in *cateringorder.php* above but for a product.

Code in *signup.php*

- `SELECT * FROM Customer WHERE Email='$email'`
 - Used to check whether submitted email in signup process matches one that is already stored in the Customer table.
- `INSERT INTO Customer(FirstName, LastName, Email, Password, Telephone) VALUES ('$firstName', '$lastName', '$email', '$password', '$telephone')`
 - Used to insert submitted customer details in the Customer table so that the customer would be registered in the database.
- `INSERT INTO Address(Email, Street, HouseNumber, Area, ZIPCode) VALUES ('$email', '$street', '$house', '$area', '$zipCode')`
 - Used to insert submitted customer address details in the Address table.

Code in *login.php*

- `SELECT * FROM Customer WHERE Email='$email' AND Password='$password'`
 - Used to check whether submitted login email and password match ones stored in the Customer table.
- `SELECT FirstName FROM Customer WHERE Email='$email'`
 - Used to retrieve the first name of customer who logged in so that it can be stored in a session variable.

Code in *product_monitoring.php* (this file is for database admin usage)

- `SELECT Product.ID, Deliverable.Name, Product.QuantityAvailable FROM Product, Deliverable WHERE Product.ID = Deliverable.ID ORDER BY Product.QuantityAvailable ASC`
 - Used to retrieve all products with their IDs and current quantity available ordered by least quantity available first, so that they can be displayed in a table for the database admin.

Code in *service_management.php* (this file is for database/website admin usage)

- `SELECT Availability FROM CateringService`
 - Used to retrieve current availability status for all catering service so that it can be viewed and changed by database or website admin.

Code in *editproduct.php* (this file is for database/website admin usage)

- `SELECT QuantityAvailable FROM Product WHERE ID = $product`
 - Used to retrieve current quantity available for chosen product so that it can be used in calculation.
- `UPDATE Product SET QuantityAvailable = $newQuantity WHERE ID = $product`

- Used to update current quantity of chosen product with quantity that database or website admin added to the old product's quantity.

Code in *editcatering.php* (this file is for database/website admin usage)

- `UPDATE CateringService SET Availability = $wedding WHERE ID = 11`
 - Updates current availability of wedding catering with new chosen availability.
- `UPDATE CateringService SET Availability = $corporate WHERE ID = 12`
 - Updates current availability of corporate catering with new chosen availability.
- `UPDATE CateringService SET Availability = $social WHERE ID = 13`
 - Updates current availability of social event catering with new chosen availability.
- `UPDATE CateringService SET Availability = $concession WHERE ID = 14`
 - Updates current availability of concession catering with new chosen availability.

5 Website Working with MySQL Database

Files in website root directory:

- **account.php**
 - php script used to access database and retrieve all of logged in customer's orders with each of their order details.
- **cateringorder.php**
 - php script that is called when logged in customer makes an order involving a catering service; accesses database to check whether chosen catering service is currently available, and if it is, inserts new order record for the customer's order appropriately in the database.
- **index.php**
- **login_page.php**
- **login.php**
 - php script that is called when customer attempts to log in; accesses database to check whether customer's submitted email and password match the ones recorded, and if they do, retrieve customer's first name from database and store it in a session variable that holds customer's info for future use in website.
- **logout_success.html**
- **logout.php**
- **order_redirect.php**
- **orderonline_page.php**
- **productorder.php**
 - php script that is called when logged in customer makes an order involving a product; accesses database to check whether chosen product quantity can be fulfilled, and if it can, inserts new order record for the customer's order appropriately in the database, then subtracts purchased product quantity from the product's available quantity in the database and appropriately updates it with the new quantity value.
- **receipt.php**
- **signup_page.html**
- **signup_success.html**
- **signup.php**

- php script that is called when customer attempts to sign up; accesses database to check whether customer has already signed up or not, and if not, inserts a new record with all of the customer's submitted details appropriately in the database.

- **style.css**

Files in admin folder (in website):

- **.htaccess**
- **.htpasswd**
- **editcatering.php**
 - php script called when database or website admin submits a change to certain catering service(s)'s current availability; accesses database to appropriately update selected catering service(s)'s current availability.
- **editproduct.php**
 - php script called when database or website admin wishes to add a quantity amount to a certain product's available quantity; accesses database to retrieve selected product's current quantity so that it can be used in calculation, then appropriately updates it with new value after addition.
- **index.html**
- **product_monitoring.php**
 - php web page that displays each product's current quantity; accesses database to retrieve each product's identifying details and current quantity.
- **service_management.php**
 - php web page that allows database or website admin to make changes to a certain product's current quantity or, one or more catering service's current availability; accesses database to retrieve each catering service's current availability for the database or website admin to see.
- **style.css**

Files in resources folder (in website)

- **background.jpg**
- **logo.png**

Steps for testing database:

1. Open XAMPP control panel and start the MySQL and Apache servers (Make sure MySQL server name is *localhost*, username is *root* and no password is set).
2. Open web browser and type in the URL "localhost/Moozys_Bakes/". Ordinarily, website folder "Moozys_Bakes" should be located in the "htdocs" folder which is in the "xampp" folder which is in the XAMPP install directory (it's important that the website folder path follows this structure in order for .htaccess file to work properly).
3. Click on the "Login/Sign Up" option in the navigation menu. Side note:
 - a. "contact" page and "products & catering services" page do not exist, they are just there in the menu for realism and will redirect you back to homepage when clicked.
 - b. You cannot access "Order" page until you have logged in.
4. Click on the "sign up" link in the bottom of the log in box.

5. Enter requested details correctly in the form (if not popup message will show up due to validation made in php side = form will not be submitted) and submit so that you can be recorded in the database.
6. After the redirect to the success message page, click on the provided link to return to website.
7. Log in with the email and password you used to sign up.
8. Click on the “Orders” option in the navigation menu and use interface to make a product order.
9. After the redirect to your receipt, click on the link in the bottom to return to website.
10. Click on the “My Account” option in navigation menu to head back to your account page, where you will see the record of the order you just made in a table.
11. Repeat step #8 but this time make a catering service order.
12. Repeat step #9 and #10, and you will see the other order you made in your account page.
13. If you wish, make any other orders until you are satisfied, and (optionally) while doing so, test for validation of input in all the input fields of both order forms.
14. Log out of your account by clicking on the log out link in the bottom of your account page.
15. Click on the “Login/Sign Up” option in the navigation menu again, and sign up a new customer.
16. Repeat steps #7 to #13 to verify that the client side (client perspective) of the database/website works well.
17. Open a new tab in the web browser.
18. Go to the admin section of the website by typing in the URL “localhost/Moozys_Bakes/admin/”
19. Enter username: *admin* and password: *1234*
 - a. If any errors occur when trying to gain access to this part of the website then it is most probably due to incorrect placement of website folder in your drive. In this case just delete the .htaccess and .htpasswd files from the admin folder.
20. Click on option #1 to see all current product quantities so far and you can verify that the quantities shown were affected correctly by the purchases you made earlier (every product has beginning quantity of 100 as that’s the way they were created in database).
21. Try to remember the quantity of the first product shown in the table (topmost product).

22. Click on the “Go Back” link then click on option #2.
23. In the “Products” section, select the product from step #21, and input a value to add to its current quantity and then press submit (The page should refresh).
24. Click on the “Go Back” link and click on option #1 again to see the change made to the product you selected.
25. Click on the “Go Back” link then go to option #2 again and this time, in the “Catering Services” section, change the availability of any catering service to “not available” and press submit (The page should refresh, and you should be able to see the availability change in the section’s table).
26. Go back to the tab that has the main website (make sure you are still logged in), and then order the catering service that you changed its availability to “not available”. The order should not go through, and a popup message should appear instead.
27. The same functionality applies to a product when its quantity reaches 0, if you want to test this as well, manually set a products available quantity to zero in the database using SQL statement and then try to order it.
28. (Optionally) you can test for validation of input for the text field in the “Products” section of option #2 (admin part of website).

6 Advanced Tasks

Conceptual/Logical relational database schema

- Converted all of EER model to conceptual/logical database schema with datatype constraints for table columns (attributes).
- Added all linkages of relations through foreign keys.

MySQL code for creating, manipulating and reading from the MySQL database

- Converted all of conceptual/logical database schema to MySQL database.
- Added more constraints appropriately to every table’s columns for optimisation ex: “not null”, “unsigned”, “unique”, “auto_increment”, “on update cascade”, “on delete cascade”.
- **1st NF:**
 - all tables have a primary key (or foreign key acting as unique key to identify each row as should be for multi-valued attributes).
 - all tables have single valued columns.
- **2nd NF:**
 - All table columns do not have partial dependency aka one primary/unique key is used to identify all values in a table row, which means all column attributes will refer only to this primary/unique key attribute. Ex: OrderID column is used to identify all corresponding order details for an order in the Orders table, and none of the order details require reference to a separate column in the table to identify it.
- **3rd NF:**

- No transitive dependencies in any of the tables. As mentioned in 2nd NF, one primary/unique key is used to identify all values in a table row, and all of these values depend on the primary/unique key.
- Complex JOIN statement embedded in php code that includes a php variable.

PHP code bridging the MySQL database with your web pages

- Many different kinds of dynamic SQL statements in php scripts.
- Added validation for user input before inserting any values in database.
- (Not really php) added .htaccess file and .htpasswd file to admin folder in order to authenticate navigation in the admin section of the website.

7 References (Optional)

- Data used to populate database with product information is taken from a selection of actual products and their prices in LE (during 2017) from the food catering business “Moozy’s Bakes” located in Egypt that belongs to my aunt, Amira Nour. The catering services data was made up by me, as the actual business doesn’t provide these kinds of services depicted in the website.
- Using .htaccess and .htpasswd files to password protect admin directory in my website:
 - <http://www.htaccesstools.com/articles/password-protection/>
- **PHP FILTER_VALIDATE_EMAIL Filter** used in **signup.php** to validate that inputted email is in fact a string that follows the format of a proper email.
 - https://www.w3schools.com/php/filter_validate_email.asp
- **PHP mysqli_real_escape_string() Function** used in most of php scripts when retrieving data of type String from forms. Used to escape special characters in string like quotation marks, tabs, entered lines, etc.
 - https://www.w3schools.com/php/func_mysqli_real_escape_string.asp
- **PHP ucfirst(), ucwords() and strtoupper() Functions** used to format inputted strings before insertion in database
 - **ucfirst()** makes first letter of a single word upper case: https://www.w3schools.com/php/func_string_ucfirst.asp
 - **ucwords()** makes first letter of each word in a string upper case: https://www.w3schools.com/php/func_string_ucwords.asp
 - **strtoupper()** makes all letters in a string upper case: https://www.w3schools.com/php/func_string_strtoupper.asp
- The **Eclipse Photon IDE for PHP Developers** was extremely helpful in providing me with all the basic mysqli and php functions needed to develop my website through suggestions and even a short explanation of each function listed in the suggestions.