Pizza Express

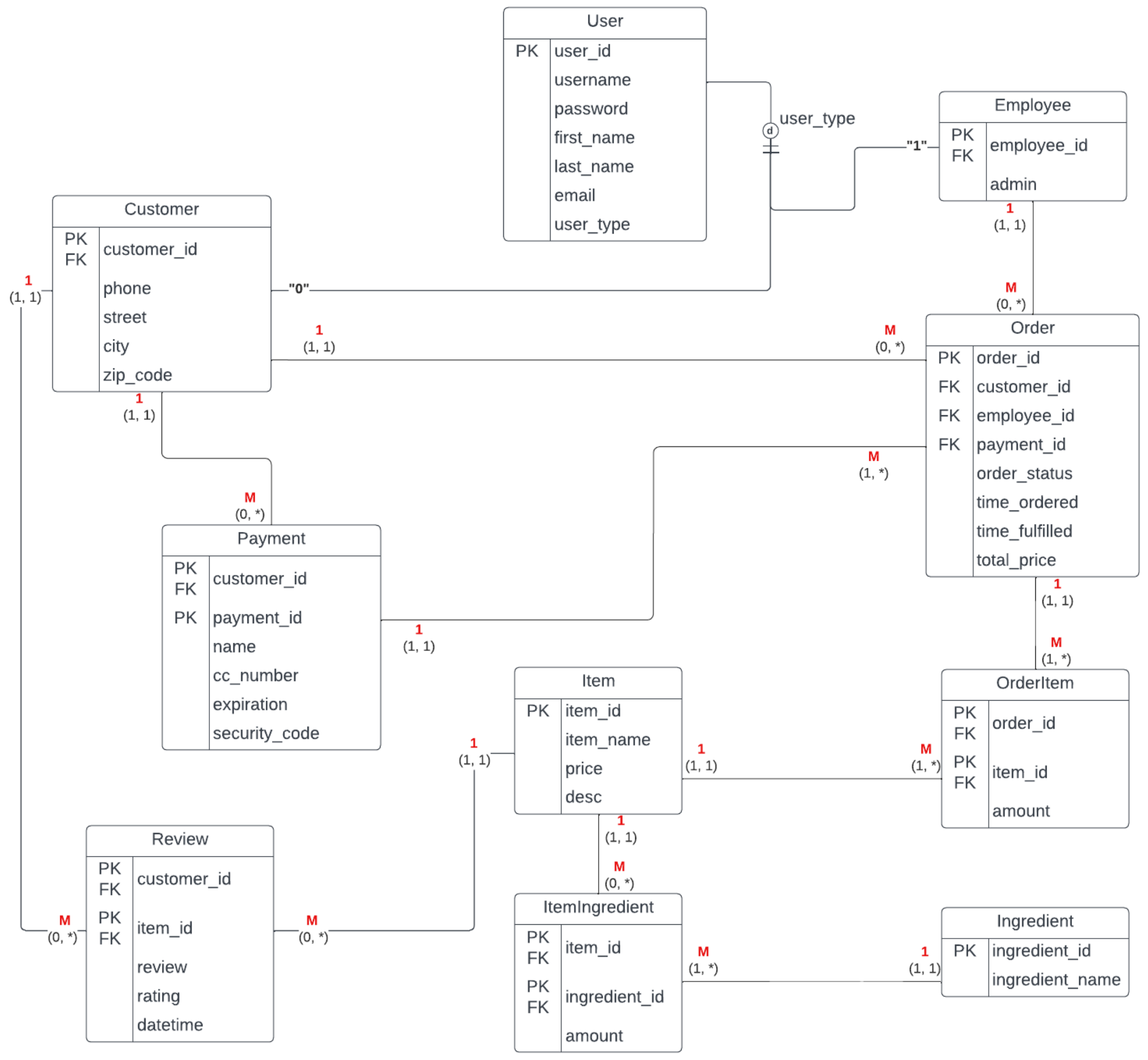
CSCE 310 - Final Project Report

Group #2:

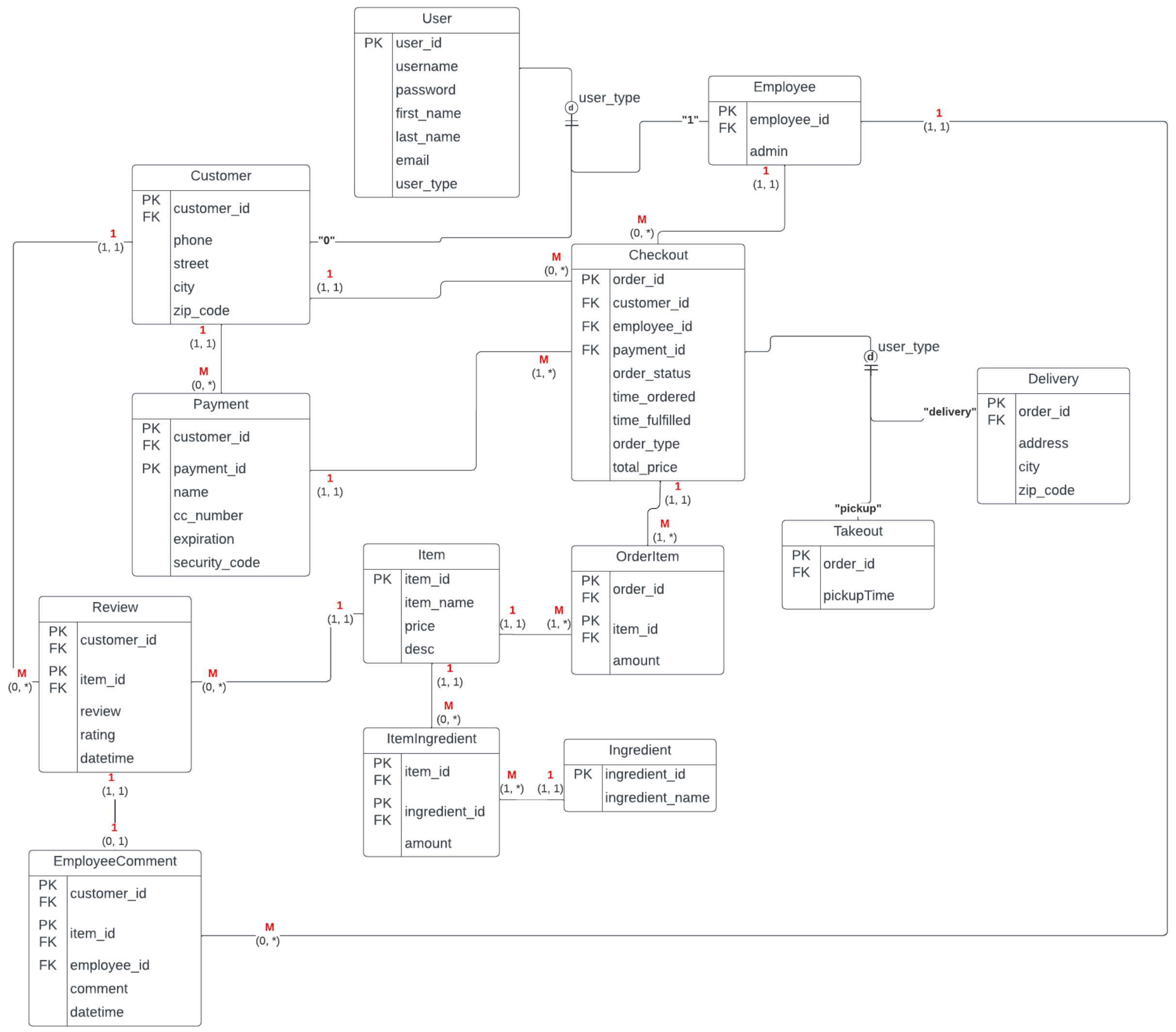
Syed Asad
Dien Chau
Ekdev Rajkitkul

Changes to Project Design (Before and After ERD):

<u>Before ERD:</u>



After ERD:



We've changed the project design in the ERD "Checkout" table (which was previously called "Order") to include subtype tables called "Takeout" and "Delivery". These would be totally disjoint so that an order would have to be either one of these types. This was needed because the two types would store different attributes.

Something we also changed in the project design was the addition of an "EmployeeComment" table, which would associate an employee comment with a user review. This would be a 1 to 1 relationship so that only one employee can leave a comment on a review, and if a new employee left a comment it would simply update the composite (customer_id, item_id) primary key with the new employee. One employee could either leave no comments or many comments.

Functionality Changes:

Dien - Dropped the functionality for “customers creating pizzas from previous history” as I thought that was more of Arjun’s role.
Arjun - Dropped the class, appointment functionality incomplete.
Syed - Cart not managed by the database but instead locally with the client. Took on extra functionality for CRUD including customer payments to fulfill customer functionality. Additionally, took on admin functionality so that an admin has CRUD operations on menu items.
Ekdev - Added functionality of filtering for reviews to facilitate use of indices.

List of all Project Files: (also include names of who completed what page):

Customer

cart.php - Syed Asad
checkout.php - Dien Chau, Syed Asad, Arjun Grover
customer_review.php - Ekdev Rajkitkul
delete_acc.php - Dien Chau
delete_review.php - Ekdev Rajkitkul
edit_account.php - Dien Chau
handle_payment_edit.php - Syed Asad
history.php - Dien Chau
home.php - Dien Chau, Arjun Grover, Syed Asad, Ekdev Rajkitkul
menu.php - Syed Asad
payment_options.php - Syed Asad
process_order.php - Dien Chau, Syed Asad, Arjun Grover
process_review.php - Ekdev Rajkitkul
review.php - Ekdev Rajkitkul
successful_order.php - Dien Chau, Arjun Grover
update_info.php - Dien Chau

Employee

add_ingredient.php - Syed Asad
add_item.php - Syed Asad
admin_menu.php - Syed Asad
comment_page.php - Ekdev Rajkitkul
delete_acc.php - Dien Chau
delete_item.php - Syed Asad
display_review.php - Ekdev Rajkitkul
edit_account.php - Dien Chau

edit_item.php - Syed Asad
employee_home.php - Dien Chau, Syed Asad, Ekdev Rajkitkul
remove_ingredient.php - Syed Asad
update_info.php - Dien Chau
update_order.php - Dien Chau

Register
create_user.php - Dien Chau, Arjun Grover
employee_register.php - Dien Chau
register.php - Dien Chau, Arjun Grover
successful_registration.php - Dien Chau, Arjun Grover

General
index.php - Dien Chau, Arjun Grover
login.php - Dien Chau, Arjun Grover
connect_db.php - Dien Chau, Arjun Grover

Functionalities:

Functionality 1: *Dien Chau*
I was responsible for the functionality set 1, which included the implementation of three different types of users: customers, employees, and admins. Admins would be a special type of employee that had extra privileges allowing them to add to the menu. I created the registration methods for the appropriate user types from the initial index.php page using an INSERT command, where a customer and employee would have different buttons to facilitate registration.

After a user is created, when a login is attempted the SELECT command is used to validate the username and password to allow access to the website, which will apply to both user types. On the home page customer and employee would have different views, where customer is able to see their ongoing orders while employee is able to see all customers orders and serve them.

UPDATE command was implemented on the edit_account page, where the respective user type could edit their account information and save their changes once done. Employees would have a small subset of the customer attributes, so both had to be updated differently.

The DELETE command had to ensure all the related appropriate data was deleted from the database. I interpreted this to mean that a customer's orders, payments, and reviews should cascade the customer deletion. On the employee side, only the employee's comments would be deleted, even though an employee would be related to an order. I believed this would be better because we wouldn't want an order to be removed from history when an employee is terminated.

For my index I did it based on the “username” attribute in the “User” table. This was because this would be used to validate a login. For my views I utilized the “customer_view” that joins the “User” table and the “Customer” table to provide a comprehensive view of both information. This was then used to autofill the text fields in the “edit_account” page.

Functionality 2: *Arjun Grover (Q-Dropped)*

Functionality 3: *Syed Asad*

My role in the project included adding commands to two user types, customer and admin, a special type of employee. I implemented an INSERT command on the admin side, where the admin is able to insert different items to the admin_menu page that are not already in the database. In addition to adding menu items, the admin is also able to add item ingredients within each menu item. On the customer side, I implemented the insert command to add payment options to pay for the orders. Therefore, the customer is able to insert multiple payment options.

The SELECT command on the customer side is implemented on the menu page. The customer is able to view the menu and select items from the menu to add to the cart. Additionally, the select command is implemented on the payment page, where the customer is able to select a payment option to pay for the order. The admin side uses the select command to recognize menu items from the database when adding, editing or deleting menu items. Whenever a menu item is typed by the user, the select command fetches that name, and implements the necessary commands by the user.

Within the customer’s payment page, the customer is able to UPDATE their payment information, by editing either the card number, security code and other relevant information. On the admin side, the user is able to update the admin menu by editing the items. The admin enters the name of the specific item that needs to be edited and then they can edit either the name, description and price of the menu item. These changes are updated on both the database and the customer view of the menu.

Another command implemented on the admin and customer side of the project was the DELETE command. On the admin side, the user is able to delete items on the admin_menu page. The user is also able to delete specific item ingredients within each menu item and save those changes. On the customer side, the customer is able to delete previously added payment options.

I added an “item_name” index in the item table that is directly reflected on the item_ingredient view to provide a detailed view of each menu item as well as ingredients within those menu items, on the admin menu page as well as the customer’s view of the menu.

Functionality 4: *Ekdev Rajkitkul*

I was responsible for functionality set 4, which spearheaded the review functionality for the customer and comment functionality for the employee/admin. Within the customer side, all four types of queries were used to allow a user to view, edit, update, and delete reviews on items. Specifically, in the page where a customer can add a review, the user_item_view was utilized to populate the dropdown on that screen with only the items the current customer has previously ordered.

Customer:
INSERT: Add review to the database with given attributes (item id, customer id, review rating, review description, datetime)

UPDATE: Update the review for an item they have already submitted a review within the database (along with attributes)

SELECT: Utilized this multiple times within my review files. Most notably used to display all customer reviews, as well as show the current customer's reviews on two separate pages from the data in the database

DELETE: Delete review from the database when a customer chooses to

Employee/Admin
As for the employee/admin side, I had implemented the employee comment functionality by forming a master page where they can view all customer reviews, along with the create/edit comment page. Within this master page, using all four types of queries along with the review_and_comment view (to populate the page with the customer review and employee comment attributes), the employee/admin can effectively post, update, and view their own comments to customer reviews, as well as delete inappropriate reviews from customers. Additionally, the rating indices were used to allow employees to filter customer reviews and employee comments by the customer rating of an item.

INSERT: Add a previously made employee comment to the database in response to a customer review, along with the attributes of customer id, item id, employee id, comment, and datetime.

UPDATE: Update an employee comment to the database in response to a customer review, along with the attributes of customer id, item id, employee id, comment, and datetime.

SELECT: Most notably used to not only view from the database the customer reviews and ratings for items, but also the associated/tag employee reviews made in response. Utilized multiple times within comment files.

DELETE: Delete any customer review from the database if deemed to be inappropriate by employee/admin.