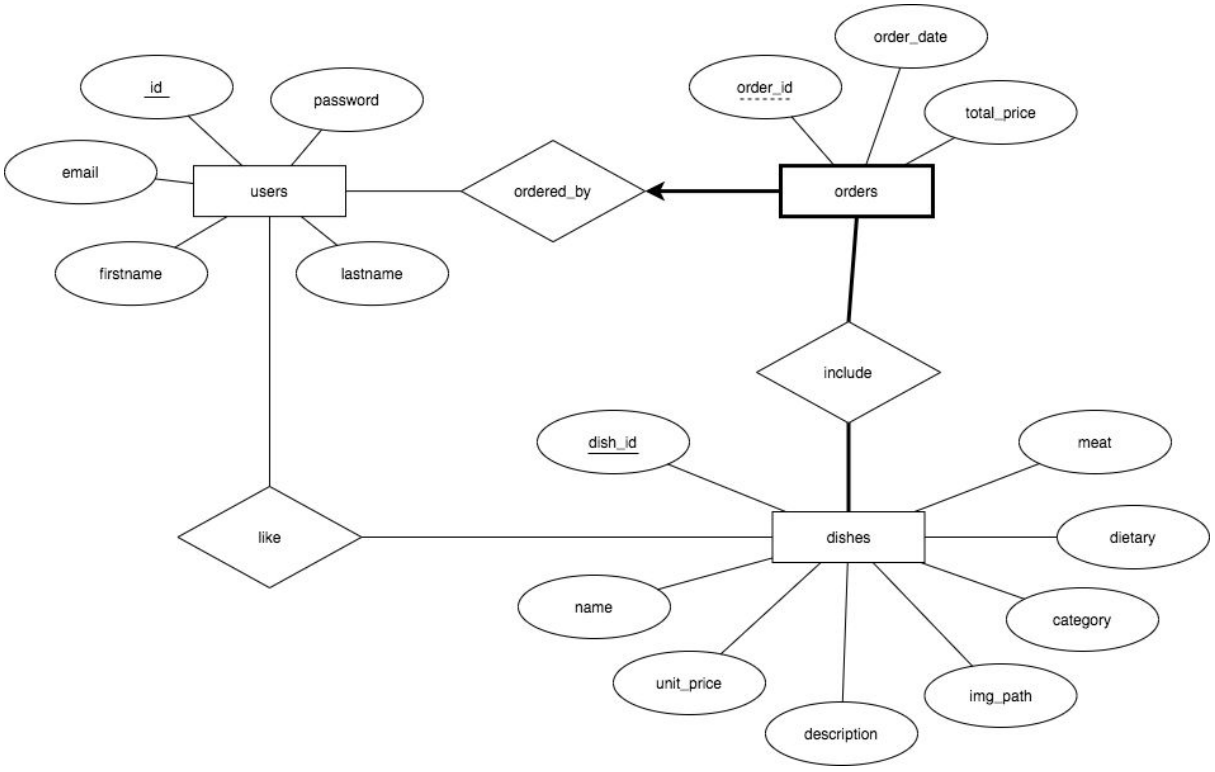
IAT 352 - PA3 Report

1. Database Design

ER Diagram:



Design Decisions:

Entities:

- Each user is identified by a unique ID (id), and also has an email, a password, first name and last name.
- Each order is identified by an order ID (order_id), and also has an order date, and total price.
- Each dish is identified by a unique ID (dish_id), and also has a name, a unit price, a brief description, a related image stored in the folder. Additionally, each dish belongs to a specific category, fits in certain dietary options, and includes one kind of meat.

Relations:

- One or more orders can be ordered by the same user, but different users won't order the same dishes at the same time.
- Each order must be ordered by at least one user.
- If a user account is deleted, all of its past orders will be removed as well.
- Each order must include at least one dish.
- One dish could be ordered multiple times in different orders.

- Users can freely mark any dish they like.
- Each dish can be liked by one or more different users.

2. Database Connection

We write the database connectivity code in a separate PHP file called connect.php so that we are able to embed this modular into any required files.

connect.php

```php
<?php

$dbhost = "localhost";
$dbuser = "root";
$dbpass = "";
$dbname = "dan_peng";

$db = mysqli_connect($dbhost, $dbuser, $dbpass, $dbname);

if(mysqli_connect_error()){
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

?>
```

Then, we use *require_once* keyword to embed connect.php into changePassword.php and manageAccount.php since these two pages need to display or change some information stored in the database.

profile.php | manageAccount.php | changePassword.php

```php
<?php
    // include auth.php file on all secure pages
    require_once("auth_sessionNotActiveCheck.php");

    // connect to database
    require_once('connect.php');
    $update_errors = array();
?>


<!DOCTYPE html>
<html>
    <head>
```

profile.php | manageAccount.php | changePassword.php

```php
<?php
    // include auth.php file on all secure pages
    require_once("auth_sessionNotActiveCheck.php");

    // connect to database
    require_once('connect.php');
    $manage_errors = array();
?>

<!DOCTYPE html>
<html>
    <head>
```

3. Secure Authentication Handling

We write the secure authentication handling code in a seperate PHP file called auth_sessionNotActiveCheck.php so that we can effectively reuse it in other files.

auth_sessionNotActiveCheck.php

```php
<?php

// check if there is an active session
// if there is no active session --> user is not logged in
// redirect user to login page
session_start();
if(!isset($_SESSION["email"])) {
  header("location: register.php");
  exit();
}

?>

```

Then, we use *require_once* keyword to embed auth_sessionNotActiveCheck.php into profile.php, manageAccount.php, and changePassword.php to make sure only members can access these pages. Otherwise, we will require them to log in or sign up first.

profile.php | manageAccount.php | changePassword.php

```php
<?php
    // include auth.php file on all secure pages
    require_once("auth_sessionNotActiveCheck.php");
?>


<!DOCTYPE html>
<html>
    <head>
```

profile.php | manageAccount.php | changePassword.php

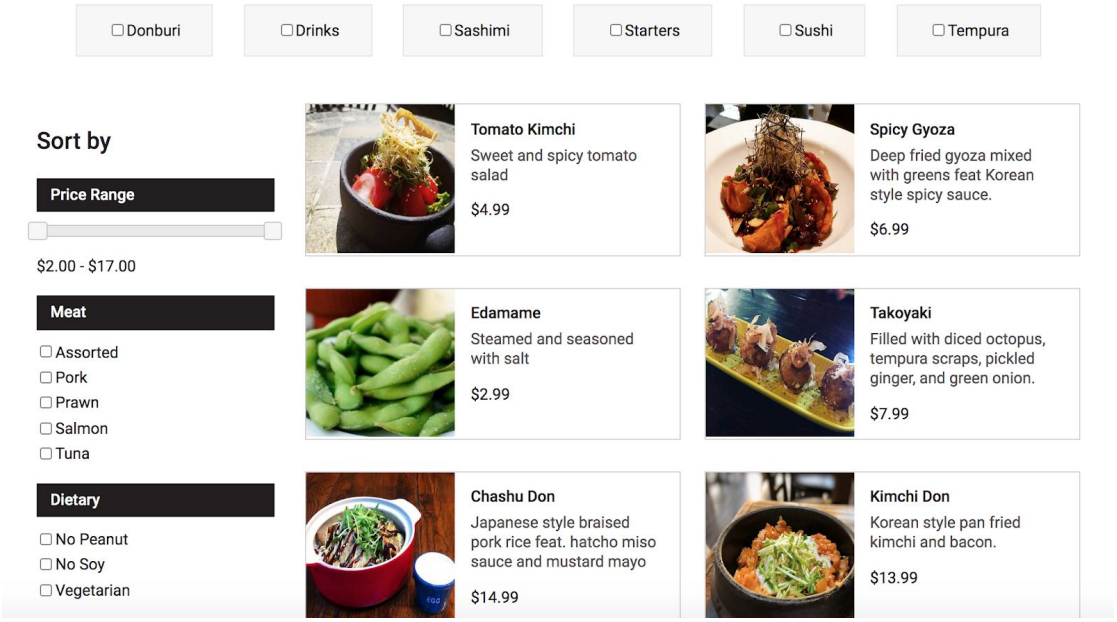
```php
<?php
    // include auth.php file on all secure pages
    require_once("auth_sessionNotActiveCheck.php");

    // connect to database
    require_once('connect.php');
    $update_errors = array();
?>


<!DOCTYPE html>
<html>
    <head>
```

profile.php | manageAccount.php | changePassword.php

```php
<?php
    // include auth.php file on all secure pages
    require_once("auth_sessionNotActiveCheck.php");

    // connect to database
    require_once('connect.php');
    $manage_errors = array();
?>

<!DOCTYPE html>
<html>
    <head>
```

4. Visitors

On the browse page, we allow visitors to browse the whole menu and apply filters to target on any dish they want to order. The menu items are saved in the database and they are all assigned into a category and labelled with some attributes, including price, which meat they include, and which type of dietary they belong to.



Code Screenshots:

```
<div class="category">
    <!-- <h2 class="filter_title">Category</h2> -->
    <?php
     $query03 = "SELECT DISTINCT category FROM dishes ORDER BY category ASC";
     $result03 = mysqli_query($db,$query03);
     while ($row = mysqli_fetch_assoc($result03)) {
        if (!empty($row['category']))
            echo "<div class='checkbox-container'>
                    <label><input type='checkbox' class='checkbox-option category category_style' value='"
                    . $row['category']. "'>"
                    . $row['category']
                . "</label></div>";
     }
    ?>
</div>
```

```
<div class="price_range">
    <h2 class="filter_title">Price Range</h2>
    <!-- A hidden field let web developers include data that can
    <input type="hidden" id="hidden_minimum_price" value="0">
    <input type="hidden" id="hidden_maximum_price" value="17">
    <div id="slider_range"></div>
    <p id="display_price">$2.00 - $17.00</p>
</div>
```

```php
<div class="meat">
    <h2 class="filter_title">Meat</h2>
    <?php
     $query01 = "SELECT DISTINCT meat FROM dishes ORDER BY meat ASC";
     $result01 = mysqli_query($db,$query01);
     while ($row = mysqli_fetch_assoc($result01)) {
        if (!empty($row['meat']))
            echo "<label><input type='checkbox' class='checkbox-option meat' value='". $row['meat']. "'>". $row['meat']. "</label>";
     }
    ?>
</div>

<div class="dietary">
    <h2 class="filter_title">Dietary</h2>
    <?php
     $query02 = "SELECT DISTINCT dietary FROM dishes ORDER BY dietary ASC";
     $result02 = mysqli_query($db,$query02);
     while ($row = mysqli_fetch_assoc($result02)) {
        if (!empty($row['dietary']))
            echo "<label><input type='checkbox' class='checkbox-option dietary' value='". $row['dietary']. "'>". $row['dietary']. "</label>";
     }
    ?>
</div>
```

We integrate HTML and PHP to create a more effective code for displaying all filters, such as category, price range, meat, and dietary. Through sending appropriate queries to the database, we can retrieve all distinct filter options and use “echo” to show them on the browser window.

5. Member Registration and Login

Visitors can register as a member by clicking the sign up button on the top right of the home page. Visitors will be guided to the registration page. When signing up, the information will be collected and stored in the database. First of all, the user's input variables are created and referenced to each of the columns in the user’s table. The second part is to check email’s format, password confirmation and whether the input fields are empty. The last part is to store the user's input into the database if there is no error found. After the account is created successfully, the user will be directed to the homepage.

Receive all inputs values from the form.

```php
$firstname = mysqli_real_escape_string($db, $_POST['first_name']);
$lastname = mysqli_real_escape_string($db, $_POST['last_name']);
$email = mysqli_real_escape_string($db, $_POST['email']);
$password_1 = mysqli_real_escape_string($db, $_POST['password_1']);
$password_2 = mysqli_real_escape_string($db, $_POST['password_2']);
```

Store receives data to the database and assigns each value to global session variables.

```php
if (count($errors) == 0) {
    $password = md5($password_1); //encrypt the password before saving in the database
    $query = "INSERT INTO users (firstname, lastname, email, password)
                VALUES('$firstname', '$lastname','$email', '$password')";
    mysqli_query($db, $query);

    $_SESSION['firstname'] = $firstname;
    $_SESSION['lastname'] = $lastname;
    $_SESSION['email'] = $email;
    $_SESSION['password'] = $password;
    $_SESSION['id'] = mysqli_insert_id($db);
    // echo $_SESSION['id'];

    header('location: index.php');
}
```

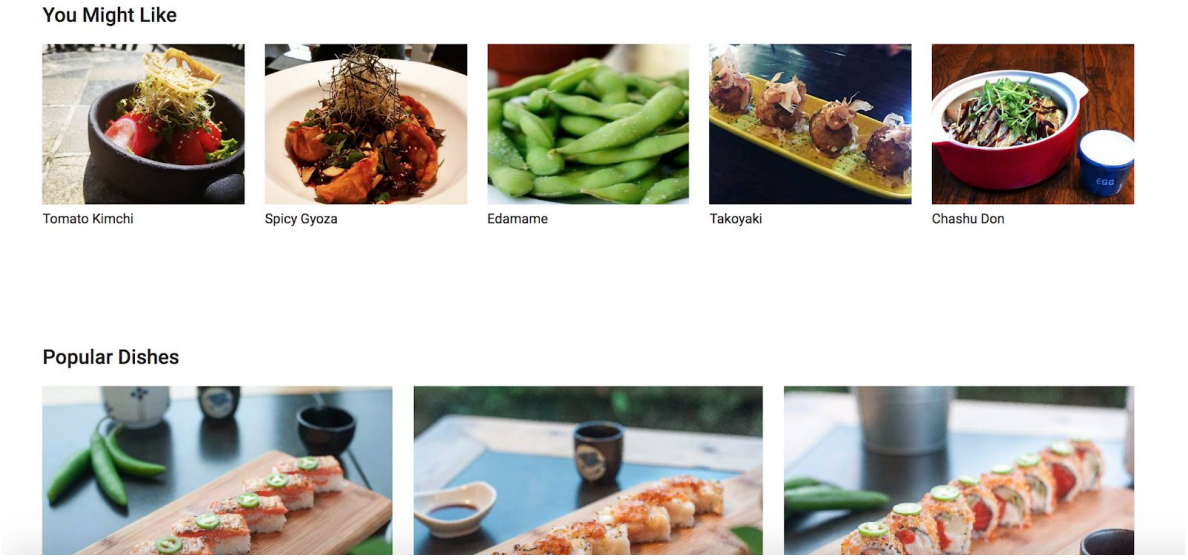
Visitors can then login as a member. Here is the code to verify if the visitor is a member.

```php
// LOGIN USER
if (isset($_POST['login_user'])) {
    $email = mysqli_real_escape_string($db, $_POST['email']);
    $password = mysqli_real_escape_string($db, $_POST['password']);

    // check if all inputs are not empty
    if (empty($email)) array_push($errors, "Email is required.");
    if (empty($password)) array_push($errors, "Password is required.");

    // check if the variable $email is a valid email address
    $check_email = filter_var($email, FILTER_VALIDATE_EMAIL);
    if ($check_email == FALSE) array_push($errors, "Please type in valid email address.");

    if (count($errors) == 0) {
        $password = md5($password);
        $query = "SELECT * FROM users WHERE email='$email' AND password='$password'";
        $results = mysqli_query($db, $query);

        if (mysqli_num_rows($results) == 1) {
            $_SESSION['email'] = $email;
            $_SESSION['password'] = $password;

            $results_array = mysqli_fetch_assoc($results);
            $_SESSION['firstname'] = $results_array['firstname'];
            $_SESSION['lastname'] = $results_array['lastname'];
            $_SESSION['id'] = $results_array['id'];

            header('location: index.php');
        } else {
            array_push($errors, "Sorry, your email or password is wrong.");
        }
    }
```

6. Personalination

On the home page, there is a section called “You Might Like” that recommends some dishes the member might like according to the dishes he/she marks as “Like”. On the contrary, there is only general content for visitors.

For members:

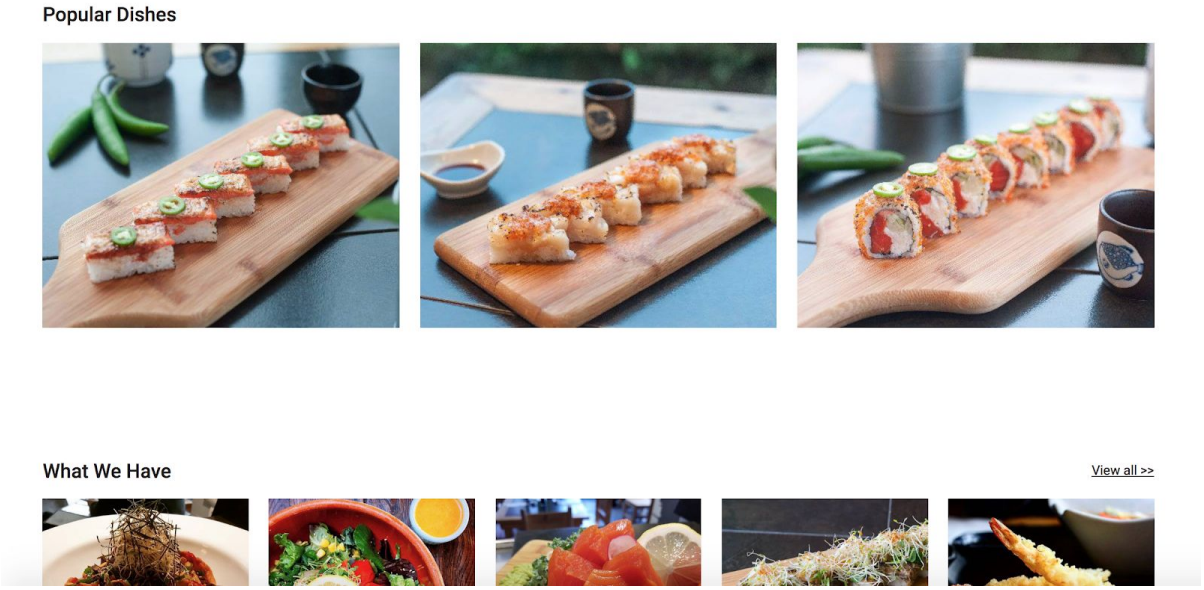


Code Screenshot:

```php
<!-- Recommendations for Registered Members -->
<?php
If (isset($_SESSION['id'])) {
    include('connect.php');
    $query = "SELECT DISTINCT dishes.img_path, dishes.name FROM dishes ";
    $query .= "INNER JOIN like_dish ON dishes.category = like_dish.category ";
    $query .= "WHERE like_dish.id = ". $_SESSION['id']. ";";
    // echo $query;
    $query_result = mysqli_query($db, $query);
    if (!$query_result) die("Database query failed.");
    $num_rows = mysqli_num_rows($query_result); // check the number of retrieved data

    if(isset($_SESSION['id']) && $num_rows > 0) {

        echo "<section><h2>You Might Like</h2><div class='container what-we-have'>";

        $count = 0;
        while ($row = mysqli_fetch_assoc($query_result)) {
            if ($count < 5) {
                echo "<figure class='item'>";
                echo "<img src='". $row['img_path']. "'>";
                echo "<figcaption class='home-category'>". $row['name']. "</figcaption>";
                echo "</figure>";
            }
            $count++;
        }
        echo "</div></section>";
        mysqli_free_result($query_result); // Release returned data
        mysqli_close($db);
    }
}
?>
```

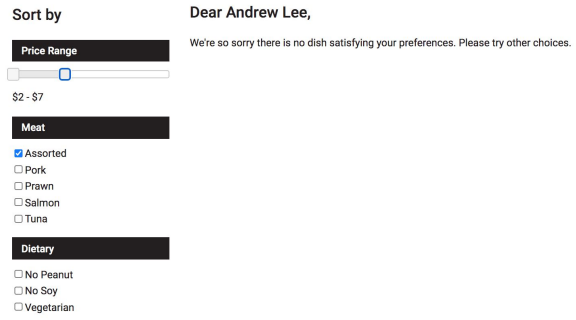
We intend to show only 5 recommendations for members on the home page because we don't want to display too much information at the beginning.

For visitors:



On the browse page, when there is no dish satisfying the preferences of the member according to checkbox selections, the website will respond with a sentence that starts with "Dear [member's name]" for members while "Dear Customer" for visitors.

For members:



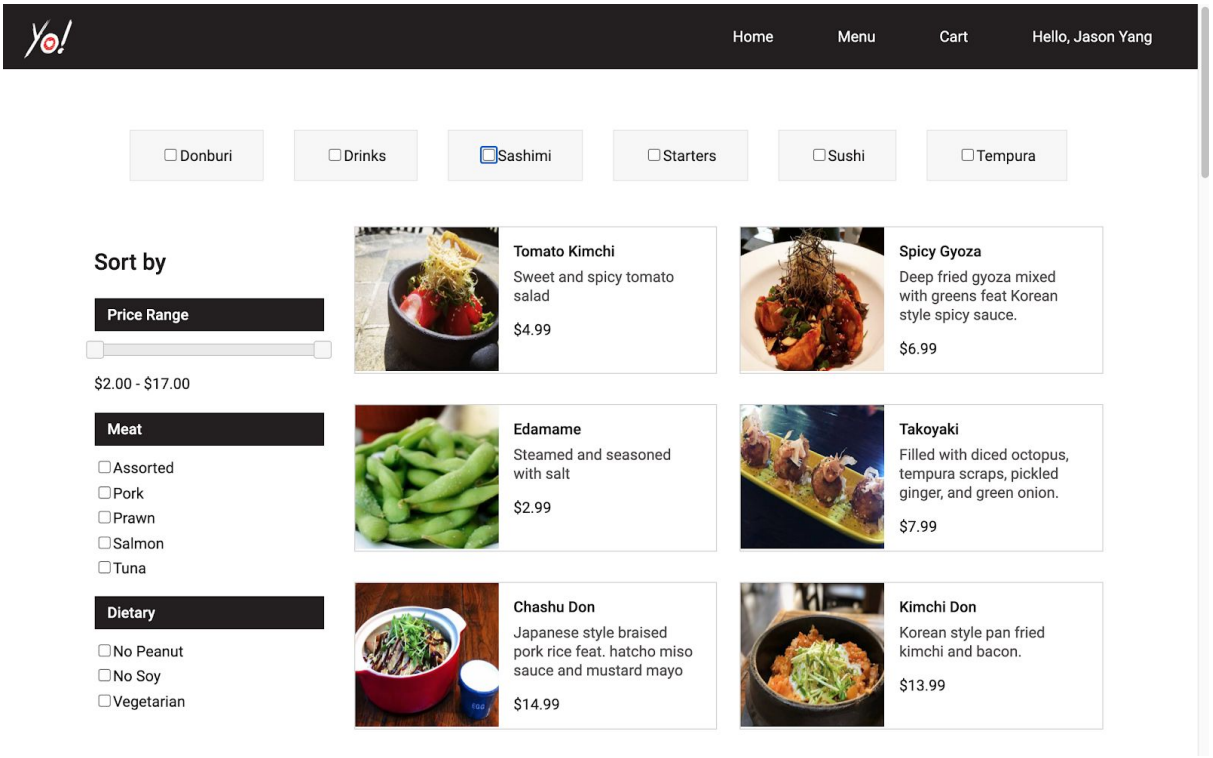
For visitors:

Code Screenshot:

```
    } else {
        $output = "<h2>Dear ";
        if (isset($_SESSION['id'])) {
            $output .= $_SESSION['firstname']." ".$_SESSION['lastname'].", ";
        } else {
            $output .= "Customer, ";
        }
        $output .= "</h2>";
        $output .= "<p>We're so sorry there is no dish satisfying your preferences. Please try other choices.</p>";
    }
    echo $output;
}
```

Here, we use stored session values to display the member name. We want to differentiate the wording for members and visitors so that our members can feel more welcome when they order food on our website.

7. Filtering Function on the Browse Page



The browse contains filter criteria such as the categories at the top and the price range, meat, and dietary types on the left column.

```php
if (isset($_POST['action'])) {

    $sql = "SELECT * FROM dishes ";
    $length = strlen($sql);

    if (isset($_POST['category'])) {
        $category_filter = implode("','", $_POST["category"]);
        if (strlen($sql) == $length) $sql .= "WHERE ";
        else $sql .= "AND ";
        $sql .= "category IN ('". $category_filter. "') ";
    }




    if (isset($_POST['minimum_price'], $_POST['maximum_price']) &&
        !empty($_POST['minimum_price']) && !empty($_POST['maximum_price'])) {
        if (strlen($sql) == $length) $sql .= "WHERE ";
        else $sql .= "AND ";
        $sql .= "unit_price BETWEEN ".$_POST["minimum_price"]." AND ".$_POST["maximum_price"]." ";
    }

    if (isset($_POST['meat'])) {
        $meat_filter = implode("','", $_POST["meat"]);
        if (strlen($sql) == $length) $sql .= "WHERE ";
        else $sql .= "AND ";
        $sql .= "meat IN ('". $meat_filter. "') ";
    }

    if (isset($_POST['dietary'])) {
        $dietary_filter = implode("','", $_POST["dietary"]);
        if (strlen($sql) == $length) $sql .= "WHERE ";
        else $sql .= "AND ";
        $sql .= "dietary IN ('". $dietary_filter. "') ";
    }
```

```php
    // echo $sql;
    $result = mysqli_query($db, $sql);
    if (!$result) die("!! Database query failed !!");

    $num_results = mysqli_num_rows($result);

    if ($num_results > 0) {
        $output = "<div class='menu_list grid'>";
        while ($row = mysqli_fetch_assoc($result)) { // a row of data for one dish
            $output .= "
            <div id='menuItem' class='menu_item'>

                <div class='item_image'>
                    <img src='".$row['img_path']."' alt='dish image'>
                </div>

                <div class='item_text'>
                    <div class='item_name' id='dish_name'>".$row['name']."</div>
                    <div class='item_description text-body-sml'>".$row['description']."</div>
                    <div class='item_price'>$".$row['unit_price']."</div>";
            // if (isset($_SESSION['id'])) {
                // $output .= "<div id='item_like'>Like</div>";
            // }
            $output .=  "</div></div>";
        }
        $output .= "</div>";
    } else {
        $output = "<h2>Dear ";
        if (isset($_SESSION['id'])) {
            $output .= $_SESSION['firstname']." ".$_SESSION['lastname'].", ";
        } else {
            $output .= "Customer, ";
        }
        $output .= "</h2>";
        $output .= "<p>We're so sorry there is no dish satisfying your preferences. Please try other choices.</p>";
    }
    echo $output;
}
```

In fetch_data.php, user's filter preferences are fetched here after the user has interacted with the filter criteria. The code in the top image allows us to get the checkbox value the user has selected.

Then, we wrote the code to display the result based on the user's filter criteria. The menu items and their details will be displayed if there is something to show, if not we then display an error message to tell them that there is no dish satisfying their preferences.

```
            <div class="category">
                <!-- <h2 class="filter_title">Category</h2> -->
                <?php
                 $query03 = "SELECT DISTINCT category FROM dishes ORDER BY category ASC";
                 $result03 = mysqli_query($db,$query03);
                 while ($row = mysqli_fetch_assoc($result03)) {
                    if (!empty($row['category']))
                        echo "<div class='checkbox-container'><label><input type='checkbox' class='checkbox-option
                            category category_style' value='". $row['category']. "'>". $row['category']. "</label></div>";
                 }
                ?>
            </div>
        </div>
    </section>
```

```
            <div class="price_range">
                <h2 class="filter_title">Price Range</h2>
                <!-- A hidden field let web developers include data that cannot be seen or modified by users when a form
                is submitted. -->
                <input type="hidden" id="hidden_minimum_price" value="0">
                <input type="hidden" id="hidden_maximum_price" value="17">
                <div id="slider_range"></div>
                <p id="display_price">$2.00 - $17.00</p>
            </div>



            <div class="meat">
                <h2 class="filter_title">Meat</h2>
                <?php
                 $query01 = "SELECT DISTINCT meat FROM dishes ORDER BY meat ASC";
                 $result01 = mysqli_query($db,$query01);
                 while ($row = mysqli_fetch_assoc($result01)) {
                    if (!empty($row['meat']))
                        echo "<label><input type='checkbox' class='checkbox-option meat' value='". $row['meat']. "'>". $
                            row['meat']. "</label>";
                 }
                ?>
            </div>

            <div class="dietary">
                <h2 class="filter_title">Dietary</h2>
                <?php
                 $query02 = "SELECT DISTINCT dietary FROM dishes ORDER BY dietary ASC";
                 $result02 = mysqli_query($db,$query02);
                 while ($row = mysqli_fetch_assoc($result02)) {
                    if (!empty($row['dietary']))
                        echo "<label><input type='checkbox' class='checkbox-option dietary' value='". $row['dietary']. "'>
                            ". $row['dietary']. "</label>";
                 }
                ?>
            </div>

        </div> <!-- end of filter_section -->
```

In browse.php, sql queries are written to display filter types and their checkboxes.

```
$(document).ready(function(){

    filter_data();


    function filter_data(){
        // $('.filter_data').html('<div id="loading"></div>');
        var action = 'fetch_data';
        var minimum_price = $('#hidden_minimum_price').val();
        var maximum_price = $('#hidden_maximum_price').val();
        var category = get_filter('category');
        var meat = get_filter('meat');
        var dietary = get_filter('dietary');
        $.ajax({
            url: "fetch_data.ph
            method: "POST",
            data: {action:action, minimum_price:minimum_price, maximum_price:maximum_price,
                category:category, meat:meat, dietary:dietary},
            success:function(data){
                $('.filter_data').html(data);
            }
        });
    }

    // Get the value of the checkbox that is selected by a user
    function get_filter(class_name){
        var filter = [];
        $('.' + class_name + ':checked').each(function(){
            filter.push($(this).val());
        });
        return filter;
    }
```

Definition:
filter.js:27

```
    // When clicking a checkbox, it will call filter_data() function:
    $('.checkbox-option').click(function(){
        filter_data();
    });


    // For price range slider
    $('#slider_range').slider({
        range: true,
        min: 2,
        max: 17,
        values: [2, 17],
        step: 0.5,
        stop:function(event, ui){
            $('#display_price').html('$' + ui.values[0] + ' - $' + ui.values[1]);
            $('#hidden_minimum_price').val(ui.values[0]);
            $('#hidden_maximum_price').val(ui.values[1]);
            filter_data();
        }
    });



});
```

In filter.js, all the filtering functions are being handled here. The filter_data function processed all the filter criterias using ajax. The get_filter function allows us to identify which box has clicked and get the value from it. Then we call the function to filter data when a checkbox or multiple checkboxes are clicked. The last of the code we determine the minimum and maximum value, as well as the scrolling value of the

price range slider. When the scroll stops, we then update the label (the min and max) at the top of the slider.

8. Personal Reflections

- Amos Xiang - Through this project, i learned how to incorporate PHP, MySQL, and AJAX together to build a functional website. There are challenges when I work on categorizing items. Since all the data are pulled from the database, I need to implement PHP and query from the database. It took me quite a lot of time to debug and figure it out. Our team was also doing a great job for we helped each other to solve the problems.

- Dan Peng - Through practicing the programming concepts with a big group project, I not only enhance my understanding of HTML, CSS, and Javascript, but I also learn PHP, MySQL, and AJAX technologies. It is exciting to first access the database and to gain basic knowledge about it. When completing the group project, the most challenging parts are the implementation of the database design with coding and the application of AJAX. It is difficult for me to simultaneously learn and apply the new knowledge within such a short time. Fortunately, with effective team collaboration, I am able to deal with the overwhelming workload and contribute to group work.

- Jason Yang - If IAT 339 provides the fundamental to front-end coding, this project enhances my ability to make a bigger and more dynamic website with the knowledge learned in back-end coding. The major challenge for me was that we had to rewrite some of the code based on the new requirements from each milestone. For example, in milestone 2, we were able to display all menu items using simple php. However in the final milestone, we had to rewrite it because we need to consider all the filtering functions. Another challenge was that sometimes changes were too slow to reflect in localhost. Overall, this project allows me to apply knowledge in PHP and MySQL to build something that is very close to a reality restaurant delivery website, which is very satisfying.