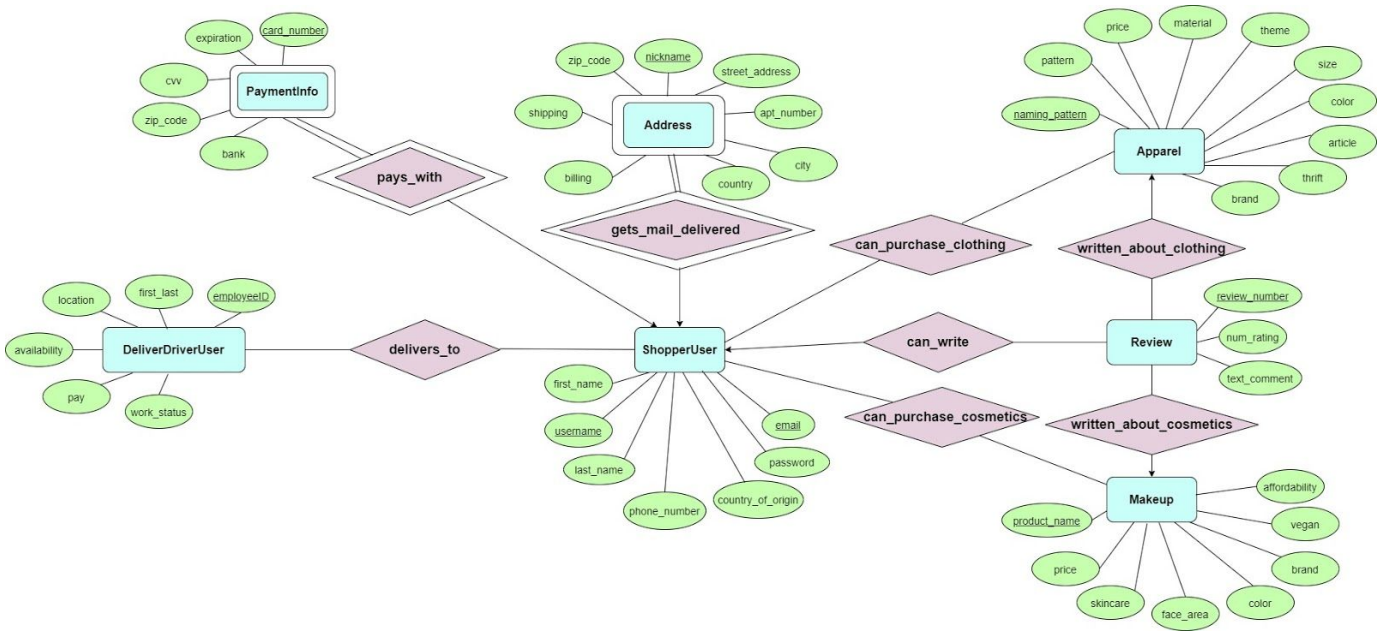
Nadia Hassan (nah3zb)
Eza Rasheed (er6qt)
Saila Salam (sbs2bt)

CS 4750 Database Systems

Final Project Report

1. Database Design:

Final Version of E-R Diagram:



Final Version of your Tables Written in Schema Statements:

Table Schema:

DeliverDriverUser(employeeID, first_last, location, pay, work_status, availability)

ShopperUser(username, email, first_name, last_name, phone_number, password, country_of_origin)
PaymentInfo(ShopperUser_username,ShopperUser_email, card_number, expiration, cvv, zip_code, bank)
Address(ShopperUser_username, ShopperUser_email, nickname, street_address, apt_number, city, country, zip_code, shipping,billing)
Apparel(naming_pattern, pattern, price, material, theme, apparel_size, color, article, thrift, brand)
Review(review_number, num_emrating, text_comment)

Makeup(product_name, price, skincare, face_area,color, brand, vegan, affordability)
delivers_to(employeeID, username, email)
can_purchase_clothing(naming_pattern, username, email)
can_purchase_cosmetics(product_name, username, email)
can_write(review_number, username, email)
written_about_cosmetics(product_name, review_number)
written_about_clothing(naming_pattern, review_number)

2. Database Programming:

- Pari's database is hosted on XAMPP's phpMyAdmin, and the app is hosted on XAMPP's localhost.
- To deploy and run the database, extract the pari.zip file to C:\xampp\htdocs. Then, start XAMPP. If there are port difficulties, configure your Apache httpd.conf - change line 59 to #Listen 12.34.56.78:8080
  and line 60 to Listen 8080
  Click XAMPP's config button → Service and Port Settings → set Main Port to 8080.
  Afterwards, open your browser and go to http://localhost:8080/phpmyadmin/ and create a new user:
  Username: pari
  Host: localhost
  Password: CS4750!!
  Check Create database with same name and grant all privileges.
  Check Grant all privileges on wildcard name (username_%).
  Check all global privileges.
  Open the file C:\xampp\htdocs\pari\PMTables.sql and copy lines 0-484. Insert the SQL into the pari phpMyAdmin database using the SQL tab and run the code to populate the web app. To view the web app, go to http://localhost:8080/pari/HomePage.php. Use the navigation bar to go through the web application tabs and various functionalities using the database that we have implemented!

- Advanced SQL Commands:

```
function implement_check(){

    global $db;
    $query = "ALTER TABLE Review ADD CHECK (num_rating<=10);";

    $statement = $db->prepare($query);
    $statement->execute();
    $results = $statement->fetchAll();
    $statement->closecursor();
    return $results;
}
```

We used this CHECK command to make sure that the review number is less than 10. If a rating above 10 is inputted, that review is not inserted into the reviews table as only a rating of 1-10 can be input.

```
function quickTrigger(){

    global $db;
    $query = "  DELIMITER $$
                CREATE TRIGGER defaultComment
                BEFORE INSERT ON Review
                FOR EACH ROW
                BEGIN
                SET new.text_comment = 'I enjoyed my experience.';
                END
                $$
                DELIMITER ;";

    $statement = $db->prepare($query);
    $statement->execute();
    $results = $statement->fetchAll();
    $statement->closecursor();
    return $results;
```

We also have a TRIGGER command to set the default comment when writing a review to “I enjoyed my experience” if it is not explicitly stated by the customer. This will act as a filler to the comments section in the reviews.

3. Database Security at the Database Level

The security is set for developers as the security of the end user's information cannot be compromised through our web application. In the following ways, we've limited access control to critical information when it comes to our users, the ShopperUsers of Pari.

- PaymentInfo
    - PaymentInfo is inputted by admins in the database through phpMyAdmin. No PaymentInfo is handled isn the web application / accessed by anyone.
- ShopperUser
    - The username and password are stored in the database and are used in order for the user to securely log in. Again, we have implemented tactics against SQL injection in order to make sure that the password the user puts cannot be breached somehow. The typed password is mapped to the database password.
- DeliveryDriverUser
    - ShopperUsers can only view information on the DeliveryDriver's name and EmployeeID but cannot view any personal information like work hours, pay, etc.
- Apparel
    - Apparel is meant to be only modified by admins to add, update, delete entries. ShopperUsers will only view the listing under the Shop Apparel tab.
- Makeup
    - Makeup listings can only be viewed, not modified, by users. Makeup is added by admins in the database through phpMyAdmin.
- Delivers_to
    - Delivers_to data can only be accessed if the user types in their username and/or email correctly. A public list of all delivers_to rows cannot be accessed in the main web app as that would be a security breach.

This was the XAMPP command used in order to create a New User Account on the Server on the phpMyAdmin localhost. The user account access credentials are as follows:

```
$username = 'pari';
$password = 'CS4750!!';
$host = 'localhost:3306';
$dbname = 'pari';
```

```
CREATE USER 'pari'@'localhost' IDENTIFIED VIA
mysql_native_password USING '***';GRANT ALL PRIVILEGES ON *.* TO
'pari'@'localhost' REQUIRE NONE WITH GRANT OPTION
MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0
MAX_UPDATES_PER_HOUR 0 MAX_USER_CONNECTIONS 0;CREATE DATABASE IF
NOT EXISTS `pari`;GRANT ALL PRIVILEGES ON `pari`.* TO
'pari'@'localhost';GRANT ALL PRIVILEGES ON `pari\_%`.* TO
'pari'@'localhost';
```

4. Database Security at the Application Level

We took several measures to ensure database security at the application level. Site owners do not have the ability to fetch or SELECT table payment information, to protect user data and the security of customers. Furthermore, to prevent the site owners from skewing reviews and ratings, pre-written reviews can only be added and viewed, not deleted or updated so as to prevent overwriting reviews. The login page has multiple levels of input validation: if either the username or password is left blank, feedback is provided to let the user know to input username and/or password. If special characters or phrases associated with SQL injections are detected, such as 'LIKE' or '%', the site throws the error "You have been reported for attempting SQL injection". For example, this code snippet was used to check that the username and password is correct, did not include SQL code, and that both were correctly provided by the user.

```
     // Check for SQL injection for username
     // admin'  AND password LIKE ' a%'
     $allClear = true;

     if (!empty($_POST['username']))
    {
     if (strpos($_POST['username'], "LIKE") !== false){
          echo  "You have been reported for attempting SQL
injection.";
          $allClear == false;
     }
     else if (strpos($_POST['username'], "%") !== false){
          echo  "You have been reported for attempting SQL
injection.";
          $allClear == false;
     }

     else if (strpos($_POST['username'], "admin' AND password
LIKE") !== false){
          echo  "You have been reported for attempting SQL
injection.";
          $allClear == false;
```

```
    }

    else if (strpos($_POST['username'], "admin' AND") !==
false){
        echo  "You have been reported for attempting SQL
injection.";
        $allClear == false;
    }

    else if (strpos($_POST['username'], "admin") !== false){
        echo  "You have been reported for attempting SQL
injection.";
        $allClear == false;
    }
  }
    // Making sure username and password were passed in
  if (empty($_POST['username'])){
      echo "Please type in a username. <br/>";
      $allClear == false;
  }
    // Password

  if (empty($_POST['password'])){
      echo "Please type in a password. <br/>";
      $allClear == false;
  }
if ((!empty($_POST['username']) && !empty($_POST['password']))
&& $found_account == true){
     if ($allClear == true)
        echo "Successfully Logged In. <br/>" ;
    }
    $found_account = false;
    $allClear = false;
```

We also have user validation for ShopperUser account creation. Pari makes sure that each field is filled, that the password and confirm password fields match, that the username is at least 4 characters, that the phone number is correctly formatted, etc. In addition, wherever the password is inputted, the input is blotched out. Password blotching is done through these snippets of code:

- The following code is used to block out the password on the ShopperUser creation page.

```
<form action="ShopperUser.php" method="post">
    <div class="form-group">
      <label for="username"> Username </label>
      <input type="text" name="username" class="form-control" />
  </div>
  <div class="form-group">
      <label for="password">Password</label>
      <input type="password" name="password" class="form-control"/>
  </div>
  <div class="form-group">
      <label for="confirmPwd">Confirm Password</label>
      <input type="password" name="confirmPwd" class="form-control"/>
  </div>
```

- The following code is used to block out the password on the Login page.

```
<form action="Login.php" method="post">
    <div class="form-group">
      <label for="username"> Username </label>
      <input type="text" name="username" class="form-control" />
   </div>
   <div class="form-group">
      <label for="password">Password</label>
      <input type="password" name="password" class="form-control"/>
   </div>

   <div class="form-group">

      <input type="submit" value="Login" class="btn btn-dark" name="db-btn" title="Insert into 'Reviews' table" />
```

5. Reflection and Peer Evaluation

- All team members completed and submitted their respective peer evaluations.
- All team members contributed fairly to this project.