P. L. A. T. F. O. R. M.

Promoting Local Advertisements to Featured Office-Room Monitors

A project by Team 13

Joaquin Merida

Abhiram Nelluri

John Liegl

Raad Chowdhury

CIS 4935 Senior Project in Information Technology - Spring 2023

**Team 13 gives consent to be an anonymous showcase or example in future classes.

Member Signatures

Abhiram Nelluri:  **Date:** 04/30/23 **25% of report**

Joaquin Merida:  **Date:** 04/30/23 **25% of report**

John Liegl:  **Date:** 04/30/23 **25% of report**

Raad Chowdhury:  **Date:** 04/30/23 **25% of report**

Summary

P.L.A.T.F.O.R.M., which stands for Promoting Local Advertisements to Featured Office-Room Monitors, is a dynamic advertising solution that can be placed in any business location with ads tailored to their clientele. For example, ads could be tailored to display specific contact lenses and corrective eye surgery options in an optometrist's office.

This "digital billboard" consists of a monitor and a Raspberry Pi that utilizes wireless internet for the retrieval of ads. The backend databases, along with the PHP code running on the site, work in unison to distribute the user-uploaded advertisements to the corresponding P.L.A.T.F.O.R.M. devices. Another feature of the P.L.A.T.F.O.R.M. system is that it will have a PIR (passive infrared) sensor to detect when people are present. This functionality will ensure two things: the screen stays off until someone approaches the device, which will help to conserve power for the businesses that choose to implement a P.L.A.T.F.O.R.M., and it will ensure that the ads are only displayed when a human is present, which will allow the backend program to utilize this information to calculate accurate metrics for analytics.

Using a web-based UI with secure client login, a user can choose to upload as many ads as their payment plan allows, and choose the specific type of business that their ad should appear in. An ad management page, which allows a user to suspend/resume an advertisement on their account, as well as view analytics regarding specific ads, are just some of the features we offer our clients.

Ads will be moderator reviewed before being published to a specific P.L.A.T.F.O.R.M. device. This is to ensure that advertisements being uploaded are suitable for the environment and audience they are hoping to reach. Once an advertisement is marked as approved, the backend system takes care of the rest and will display the ad on the appropriate devices until the user's ad plan expires. If an advertisement is declined, for whatever reason, a note will be sent to the user who uploaded the ad informing them of the decision and allowing them to re-submit an ad that fits the necessary requirements.

Support

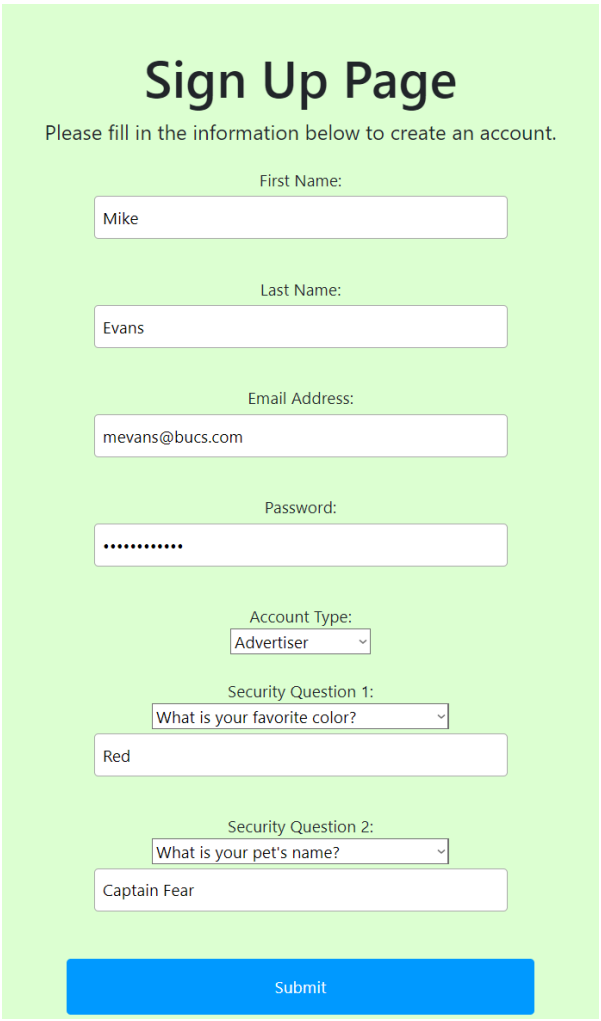
No outside support was provided on this project. The project was completed solely with each member’s prior knowledge and knowledge gained through research and hands-on experience with this project.

Functional Requirements Checklist

Requirement 1: The web UI shall require new users to create a username, password, and complete contact information.
Competition Rate: 100%
Implementer/Tester: Abhi (Username and Password), John (Contact/Billing Info)

Username and Password

Here is the page that prompts the user for the required information:



Here is the HTML code for this page:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Account Setup</title>
    <script src="signup.js"></script>

        <!-- Bootstrap CSS Needed for Navbar, etc. -->
        <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.
css" rel="stylesheet"
integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65Vohhpuu
COmLASjC" crossorigin="anonymous">
```

```
        <!-- Bootstrap Icons -->
        <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-ic
ons.css">

        <!-- Javascript source for Navbar import code-->
        <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></s
cript>

        <!-- References the main.css file for style alterations -->
      <link rel="stylesheet" type="text/css" href="main.css" id="csslink">

        <!-- Makes the title the name of the tab -->
      <title>Account Setup</title>
    <style>
      /* Centering everything except the logo */
      /* JAY'S NOTE: Commented the body tag selector out as it was messing
up the navbar and logo.
                  Removal didn't affect any other element on page. */
      /* body {
      display: flex;
      flex-direction: column;
      align-items: center;
      } */

      .container {
      margin-top: 50px;
      text-align: center;
      }

      form {
      margin-top: 20px;
      display: flex;
      flex-direction: column;
      align-items: center;
      }

      /* Styling the logo */
```

```
      .imgCenter {
      display: block;
      margin-left: auto;
      margin-right: auto;
      max-width: 100%;
      height: auto;
      }

      /* Styling the requirements */
      .requirements {
      color: red;
      display: block;
      }

      .fulfilled {
      color: green;
      display: block;
      }
    </style>

<link rel="stylesheet" href="theme.css" type="text/css" />
<script src="theme.js"></script>

</head>
<!-- On refresh the page goes to the saved theme option -->
  <!-- Navbar section -->
  <nav id="nav-placeholder">
  </nav>
  <script>
      $.get("navigation.html", function(data){
          $("#nav-placeholder").replaceWith(data);
        });
    </script>
  <!-- End Navbar section -->

  <body onload="currenttheme()">
    <!-- Displays the logo of PLATFORM -->
    <br><div style="text-align: center;">
      <img src="logo.png" alt="logo" class="imgCenter">
    </div>
```

```
    <!-- Displays the title of the page and instructions  -->
    <div class="container">
      <h1 class="size"><br>Sign Up Page</h1>
      <p>Please fill in the information below to create an account.</p>
    </div>

    <form method="post" action="setup-db.php" onsubmit="return
validateForm()">
      <label for="first-name">First Name:</label>
      <input type="text" id="first-name" name="first-name"><br>

      <label for="last-name">Last Name:</label>
      <input type="text" id="last-name" name="last-name"><br>

      <label for="username">Email Address:</label>
      <input type="email" id="username" name="username"><br>

      <label for="password">Password:</label>
      <input type="password" id="password" name="password"><br>

      <label for="account-type">Account Type:</label>
      <select id="account-type" name="account-type">
        <option value="1">Moderator</option>
        <option value="2">Advertiser</option>
        <option value="3">Business Owner</option>
      </select><br>

      <label for="security-question-1">Security Question 1:</label>
      <select id="security-question-1" name="security-question-1"
required>
        <option value="" disabled selected>Select a question</option>
        <option value="1">What is your mother's maiden name?</option>
        <option value="2">What is your favorite color?</option>
        <option value="3">What is your favorite food?</option>
        <option value="4">What is your pet's name?</option>
      </select>
      <input type="text" id="security-answer-1"
name="security-answer-1"><br>
```

```
      <label for="security-question-2">Security Question 2:</label>
      <select id="security-question-2" name="security-question-2"
required>
        <option value="" disabled selected>Select a question</option>
        <option value="1">What is your mother's maiden name?</option>
        <option value="2">What is your favorite color?</option>
        <option value="3">What is your favorite food?</option>
        <option value="4">What is your pet's name?</option>
      </select>
      <input type="text" id="security-answer-2"
name="security-answer-2"><br>

      <input type="submit" value="Submit">
    </form>

    <!-- Bootstrap JavaScript needed for some components to function -
Embed as last thing in body of HTML. -->
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle
.min.js"
integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/t
WtIaxVXM" crossorigin="anonymous"></script>

  </body>
</html>
```

Here is the PHP code for this page (functions as the backend connection to the database and inserts all the user-given values into the database):

```
<?php
    ini_set('display_errors', 1);
    ini_set('display_startup_errors', 1);
    error_reporting(E_ALL);


    include "connect.php";
?>

<!DOCTYPE html>
<html>
```

```
<head>
    <meta charset="utf-8">
</head>
<body>

<?php
    // Check if the form has been submitted
    if (isset($_POST) && !empty($_POST)) {
        // Get form data
        $firstName = $_POST['first-name'];
        $lastName = $_POST['last-name'];
        $email = $_POST['username'];
        $password = $_POST['password'];
        $passwordHash = password_hash($password, PASSWORD_DEFAULT);
        $acctTypeID = intval($_POST['account-type']);
        $secQuesOneID = intval($_POST['security-question-1']);
        $answer1 = $_POST['security-answer-1'];
        $question1Hash = password_hash($answer1, PASSWORD_DEFAULT);
        $secQuesTwoID = intval($_POST['security-question-2']);
        $answer2 = $_POST['security-answer-2'];
        $question2Hash = password_hash($answer2, PASSWORD_DEFAULT);

/*         echo 'Account Type ID: ' . $acctTypeID . '<br>';
        echo 'Security Question 1 ID: ' . $secQuesOneID . '<br>';
        echo 'Security Answer 1: ' . $answer1 . '<br>';
        echo 'Security Answer 1 HASH: ' . $question1Hash . '<br>';
        echo 'Security Question 2 ID: ' . $secQuesTwoID . '<br>';
        echo 'Security Answer 2 : ' . $answer2 . '<br>';
        echo 'Security Answer 2 HASH : ' . $question2Hash . '<br>';
        exit(); */

        // Insert user data
        $sql = "INSERT INTO User (firstName, lastName, email, acctTypeID,
secQuesOneID, secQuesTwoID)
                VALUES ('$firstName', '$lastName', '$email',
'$acctTypeID', '$secQuesOneID', '$secQuesTwoID')";
        $result1 = mysqli_query($conn, $sql);

        // Get the user ID of the newly inserted user
        $userID = mysqli_insert_id($conn);
```

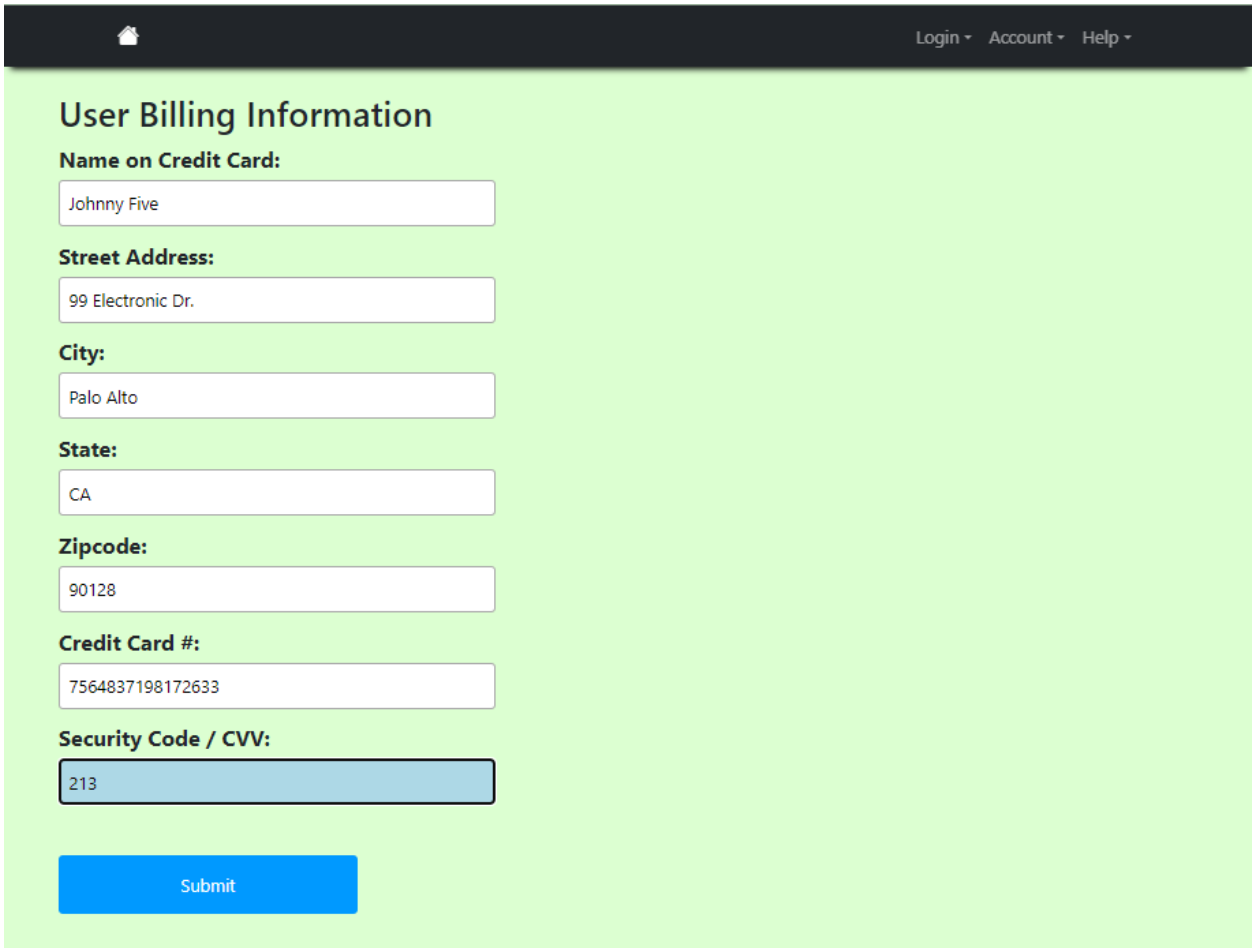
```
        // Insert password hash
        $sql = "INSERT INTO UserPassword (userID, passwordHash) VALUES
('$userID', '$passwordHash');";
        $result2 = mysqli_query($conn, $sql);

        // Insert security question answers
        $sql = "INSERT INTO SecurityQuestionAnswers (userID,
secQuestionID, questionHash)
                VALUES ('$userID', '$secQuesOneID', '$question1Hash'),
                ('$userID', '$secQuesTwoID', '$question2Hash');";
        $result3 = mysqli_query($conn, $sql);

        $conn->close();

        // Redirect to login webpage
        header("Location: login.html");
        exit();
    }
?>
</body>
</html>
```

Here are the User and UserPassword tables in the database after clicking the Submit button (observe userID = 35):

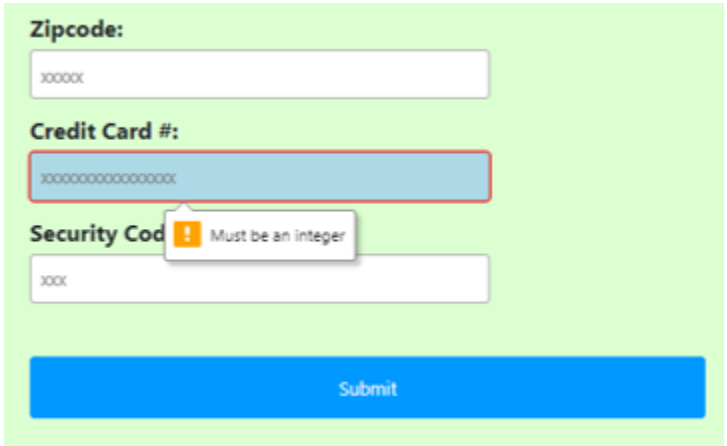
```
MariaDB [p2]> select * from User;
+--------+-----------+-----------+-------------------+------------+--------------+--------------+-----------+--------+----------------+---------+------------+
| userID | firstName | lastName  | email             | acctTypeID | secQuesOneID | secQuesTwoID | paymentID | planID | numOfAdsOnAcct | newUser | acctLocked |
+--------+-----------+-----------+-------------------+------------+--------------+--------------+-----------+--------+----------------+---------+------------+
|      1 | Joaquin   | Merida    | jmerida@usf.edu   |          1 |            2 |            3 |      NULL |      3 |              9 |       1 |          0 |
|      2 | John      | Liegl     | jliegl@usf.edu    |          1 |            3 |            4 |      NULL |      1 |              0 |       1 |          0 |
|      3 | Abhiram   | Nelluri   | anelluri@usf.edu  |          1 |            1 |            2 |      NULL |      0 |              0 |       1 |          0 |
|      4 | Bob       | Bob       | bob@bob.com       |          3 |            1 |            4 |      NULL |      0 |              0 |       1 |          0 |
|      5 | Marley    | Marley    | Marley@bob.com    |          1 |            1 |            3 |      NULL |      0 |              0 |       1 |          0 |
|      6 | Mario     | Mario     | mario@bob.com     |          3 |            2 |            4 |      NULL |      0 |              0 |       1 |          1 |
|      7 | FinalTest | FinalTest | FinTest@bob.com   |          2 |            1 |            4 |      NULL |      0 |              0 |       1 |          1 |
|      8 | Abraham   | Lincoln   | lincoln@bob.com   |          3 |            4 |            1 |      NULL |      0 |              0 |       1 |          0 |
|     17 | Tom       | Brady     | tBrady@gmail.com  |          2 |            2 |            4 |      NULL |      0 |              0 |       1 |          1 |
|     20 | Randy     | Moss      | rMoss@gmail.com   |          3 |            1 |            3 |      NULL |      0 |              0 |       1 |          0 |
|     22 | Tyreek    | Hill      | tHill@gmail.com   |          1 |            1 |            4 |      NULL |      0 |              0 |       1 |          0 |
|     24 | Peter     | Merida    | pete@usf.edu      |          1 |            2 |            4 |        12 |      1 |              0 |       1 |          0 |
|     25 | Panda     | Phantom   | broadsIn@tl.edu   |          1 |            1 |            3 |      NULL |      0 |              0 |       1 |          0 |
|     26 | Drew      | Brees     | brees@saints.com  |          3 |            2 |            3 |      NULL |      0 |              0 |       1 |          0 |
|     27 | Guy       | Fieiri    | flavor@town.com   |          2 |            2 |            4 |      NULL |      0 |              7 |       1 |          0 |
|     28 | Bobby     | Smith     | email@gmail.com   |          2 |            1 |            2 |      NULL |      0 |              0 |       1 |          0 |
|     29 | Yaz       | Yaz       | Yaz@yaz.com       |          2 |            1 |            2 |        17 |      2 |              0 |       1 |          0 |
|     30 | Walter    | White     | email@gmail.com   |          2 |            1 |            2 |      NULL |      3 |              0 |       1 |          0 |
|     31 | SpongeBob | Square    | krusty@gmail.com  |          2 |            1 |            4 |      NULL |      3 |              0 |       1 |          0 |
|     32 | Eugene    | Krabs     | krabs@gmail.com   |          2 |            2 |            3 |        14 |      3 |              0 |       1 |          0 |
|     33 | Clark     | Kent      | superman@gmail.com |          2 |            1 |            2 |        15 |      3 |              0 |       1 |          0 |
|     34 | Bruce     | Wayne     | batman@gmail.com  |          1 |            1 |            2 |        16 |      3 |              1 |       1 |          0 |
|     35 | Mike      | Evans     | mevans@bucs.com   |          2 |            2 |            4 |      NULL |      0 |              0 |       1 |          0 |
+--------+-----------+-----------+-------------------+------------+--------------+--------------+-----------+--------+----------------+---------+------------+
23 rows in set (0.000 sec)

MariaDB [p2]> select * from UserPassword;
+--------+--------------------------------------------------------------+
| userID | passwordHash                                                 |
+--------+--------------------------------------------------------------+
|      1 | $2y$10$J5IU0892WV5V3j72kE7xjOUDogYIsnr/ddQTBVvI.CvpgVs68kbwK |
|      4 | $2y$10$J5IU0892WV5V3j72kE7xjOUDogYIsnr/ddQTBVvI.CvpgVs68kbwK |
|      5 | LifeSucks1234%                                               |
|      6 | Mario01234%                                                  |
|      7 | $2y$10$rJzOf3mFi6zEfMdwoeWOl.S/cJbA9xs9tDfwlLUEtAWyOIyOBoPpu |
|      8 | $2y$10$Zy2P1ZGZafL3tF1jXhsGbONGJ1FeIYNmbIGTRmvcmjYDP3plZgSf6 |
|      2 | J123                                                         |
|     17 | $2y$10$Mi6ph9lQMDMmk1vPAFVCuuZ0KBkJ6j.OW5X5aCQgusZ6eTeMuHlk2 |
|     20 | $2y$10$R4oP30HCJYdZYL/0yUsKsOUzexOMs1HXYeUiFhc7HMPh0/GAs3xsq |
|     22 | $2y$10$wz6cKlyP5Yt9QVqJu/fK/e4gXsewe7H2bruDPjwsxqKHKHXI3BTXm |
|     24 | $2y$10$Upaa02hj257Gosax/uCs2eIsOcGLYa./Uy33LkUmraNJUOIuKpTi6 |
|     25 | $2y$10$J5IU0892WV5V3j72kE7xjOUDogYIsnr/ddQTBVvI.CvpgVs68kbwK |
|     26 | $2y$10$oO4UnazMe2pSpZQ2Ck2DJOlZqW9nvqBa7hWDg8KfAL/AoPbeuiv/e |
|     27 | $2y$10$J5IU0892WV5V3j72kE7xjOUDogYIsnr/ddQTBVvI.CvpgVs68kbwK |
|     28 | $2y$10$BUeXHllUz9.kV9Eh6DsDf.eaAQD6tpRxnanK0qmDsQq72sYEZGyyO |
|     29 | $2y$10$WGR/tUEY3HqCIFaevsYMAelqBDdBBpRMgoGbPE3sTszZRFNP8a.d. |
|     30 | $2y$10$/o5lUcFfTSWZPwYIf3QEFepf1qAKR7rsAl6siyQg74PU.OY2zy346 |
|     31 | $2y$10$KrfIqTDrArMyENL7jiEhKeUCTs/lPjW/KDrAmTxHlEapNYZFcsj3S |
|     32 | $2y$10$GQGtRYwQGIr8CX10ksK2CuTKlLnkOd3HtmVzSvGMpV6VdKwD18iKy |
|     33 | $2y$10$fRcIAX1IjjIEun8eJm7douQ6kvMVScWtZ3PzRqofS9ngBPscyDaAS |
|     34 | $2y$10$MX2hzKbvQeiONZNJ.mZsLeE.3C2ZrOKheX8gxHgP6cf73RMgjhZXu |
|     35 | $2y$10$h6.JLrEPuXdFAjiDP3TiBOieylkSLPxtJ1.XI7I5gnpKofOBU8vkW |
+--------+--------------------------------------------------------------+
22 rows in set (0.000 sec)
```

Contact Information

On the contact/billing information screen below we can see that the user has entered his information according to what has been requested.



As an example an error is thrown if the user attempts to enter anything other than numbers in fields asking only for numbers. This applies to the CVV, Zip Code, and Credit Card number fields.



Here are the database tables for Addresses and PaymentInfo before adding in Johnny Five's information above.

```
MariaDB [p2]> SELECT * FROM Addresses;
+-----------+---------------------+---------------+-------+---------+
| addressID | address             | city          | state | zipCode |
+-----------+---------------------+---------------+-------+---------+
|         1 | 1234 SW Database Dr. | Tampa         | FL    | 33615   |
|         2 | 66 High Ground Hill  | Mustafar      | NM    | 23232   |
|         3 | 5678 Infosec Ave.    | Valrico       | FL    | 33594   |
|         4 | 3498 Lois Ln.        | Metropolis    | NY    | 45872   |
|         5 | 2345 Duder Dr.       | St. Petersburg | FL    | 34231   |
|         7 | 123 High Hill Rd.    | Phoenix       | AZ    | 78739   |
|         8 | 456 Traction Dr.     | Omaha         | NE    | 55784   |
|         9 | 12345 ASDFSA         | Miami         | OK    | 73099   |
|        10 | 785 Cheers Ave.      | Green Bay     | WI    | 58729   |
|        11 | 12345  asdfgh        | Yukon         | OK    | 73099   |
|        12 | Order 66 Dr.         | Tattooine     | CO    | 12345   |
+-----------+---------------------+---------------+-------+---------+
11 rows in set (0.000 sec)

MariaDB [p2]> SELECT * FROM PaymentInfo;
+-----------+------------------+--------------------+-----+-----------+
| paymentID | ccNumber         | nameOnCard         | cvv | addressID |
+-----------+------------------+--------------------+-----+-----------+
|         1 | 2345234523452345 | Jim Jones          | 123 |         1 |
|         2 | 0987654321123456 | Joaquin Merida     | 342 |         4 |
|         3 | 8765123490903456 | John Liegl         | 999 |         2 |
|         4 | 1111222233334444 | Joey McFadden      | 087 |         5 |
|         5 | 9988776655443322 | Joey McFadden      | 111 |         5 |
|         7 | 5555666677778888 | Billy Idol         | 777 |         5 |
|         8 | 1234123412341234 | Joaquin Test Three | 356 |         8 |
|         9 | 1234123412341234 | Joaquin Test Four  | 908 |         9 |
|        10 | 1234522345761823 | Ted Danson         | 567 |        10 |
|        11 | 1234567899876543 | Test Five          | 123 |        11 |
|        12 | 8008135696969696 | Pete Merida        | 777 |        12 |
+-----------+------------------+--------------------+-----+-----------+
11 rows in set (0.000 sec)
```

And here those same tables are after the user adds in their new address and billing information for their account.

```
MariaDB [p2]> SELECT * FROM Addresses;
+-----------+----------------------+----------------+-------+---------+
| addressID | address              | city           | state | zipCode |
+-----------+----------------------+----------------+-------+---------+
|         1 | 1234 SW Database Dr. | Tampa          | FL    | 33615   |
|         2 | 66 High Ground Hill  | Mustafar       | NM    | 23232   |
|         3 | 5678 Infosec Ave.    | Valrico        | FL    | 33594   |
|         4 | 3498 Lois Ln.        | Metropolis     | NY    | 45872   |
|         5 | 2345 Duder Dr.       | St. Petersburg | FL    | 34231   |
|         7 | 123 High Hill Rd.    | Phoenix        | AZ    | 78739   |
|         8 | 456 Traction Dr.     | Omaha          | NE    | 55784   |
|         9 | 12345 ASDFSA         | Miami          | OK    | 73099   |
|        10 | 785 Cheers Ave.      | Green Bay      | WI    | 58729   |
|        11 | 12345  asdfgh        | Yukon          | OK    | 73099   |
|        12 | Order 66 Dr.         | Tattooine      | CO    | 12345   |
|        13 | 99 Electronic Dr.    | Palo Alto      | CA    | 90128   |
+-----------+----------------------+----------------+-------+---------+
12 rows in set (0.000 sec)

MariaDB [p2]> SELECT * FROM PaymentInfo;
+-----------+------------------+--------------------+-----+-----------+
| paymentID | ccNumber         | nameOnCard         | cvv | addressID |
+-----------+------------------+--------------------+-----+-----------+
|         1 | 2345234523452345 | Jim Jones          | 123 |         1 |
|         2 | 0987654321123456 | Joaquin Merida     | 342 |         4 |
|         3 | 8765123490903456 | John Liegl         | 999 |         2 |
|         4 | 1111222233334444 | Joey McFadden      | 087 |         5 |
|         5 | 9988776655443322 | Joey McFadden      | 111 |         5 |
|         7 | 5555666677778888 | Billy Idol         | 777 |         5 |
|         8 | 1234123412341234 | Joaquin Test Three | 356 |         8 |
|         9 | 1234123412341234 | Joaquin Test Four  | 908 |         9 |
|        10 | 1234522345761823 | Ted Danson         | 567 |        10 |
|        11 | 1234567899876543 | Test Five          | 123 |        11 |
|        12 | 8008135696969696 | Pete Merida        | 777 |        12 |
|        13 | 7564837198172633 | Johnny Five        | 213 |        13 |
+-----------+------------------+--------------------+-----+-----------+
12 rows in set (0.000 sec)
```

Here is the code behind the HTML page for the Profile/Billing Management page and sends user entered data to the update_billing_DB.php page for handling.

```
<h2>User Billing Information</h2>
<form action="update_billing_DB.php" method="POST">
    <label class="label" for="ccname">Name on Credit Card:</label><br>
    <input type="text" id="ccname" name="ccname"><br>

    <label class="label" for="address">Street Address:</label><br>
    <input type="text" id="address" name="address"><br>

    <label class="label" for="city">City:</label><br>
    <input type="text" id="city" name="city"><br>

    <label class="label" for="state">State:</label><br>
    <input type="text" id="state" name="state" value="" placeholder="XX" maxlength="2"><br>

    <label class="label" for="zip">Zipcode:</label><br>
    <input type="text" id="billZip" name="zip" value="" maxlength="5" autocomplete="postal-code" placeholder="xxxxx" pattern="^([0

    <label class="label" for="crednum">Credit Card #:</label><br>
    <input type="tel" id="crednum" name="crednum" inputmode="numeric" pattern="[0-9\s]{16,19}" autocomplete="cc-number" maxlength=

    <label class="label" for="cvv">Security Code / CVV:</label><br>
    <input type="text" id="cvv" name="cvv" maxlength="3" autocomplete="cc-csc" placeholder="xxx" pattern="^([0-9]{3})$"><br><br>

    <input type="submit" value="Submit">
</form>
```

Here is the script for the HTML page that manages the text entry requirements, keeping users from entering incorrect or dangerous data into the text fields.

```
<script>
    // Restricts input for the given textbox to the given inputFilter.
    function setInputFilter(textbox, inputFilter, errMsg) {
      ["input", "keydown", "keyup", "mousedown", "mouseup", "select", "contextmenu", "drop", "focusout"].forEach(function(event) {
        textbox.addEventListener(event, function(e) {
          if (inputFilter(this.value)) {
            // Accepted value - character entered matches set allowed by "setInputFilter".
            if (["keydown","mousedown","focusout"].indexOf(e.type) >= 0){
              this.classList.remove("input-error");
              this.setCustomValidity("");
            }
            this.oldValue = this.value;
            this.oldSelectionStart = this.selectionStart;
            this.oldSelectionEnd = this.selectionEnd;
          } else if (this.hasOwnProperty("oldValue")) {
            // Rejected value - restore the previous saved value - happens when data already in field and character not in "setInp
            this.classList.add("input-error");
            this.setCustomValidity(errMsg);
            this.reportValidity();
            this.value = this.oldValue;
            this.setSelectionRange(this.oldSelectionStart, this.oldSelectionEnd);
          } else {
            // Rejected value - nothing to restore as field was blank
            this.value = "";
          }
        });
      });
    }


    setInputFilter(document.getElementById("billZip"), function(value) {
      return /^\d*$/.test(value); }, "Must be an integer");
    setInputFilter(document.getElementById("crednum"), function(value) {
      return /^\d*$/.test(value); }, "Must be an integer");
    setInputFilter(document.getElementById("cvv"), function(value) {
      return /^\d*$/.test(value); }, "Must be an integer");
    setInputFilter(document.getElementById("ccname"), function(value) {
      return /^[a-z\s.]*$/i.test(value); }, "Must use alphabetic latin characters");
    setInputFilter(document.getElementById("city"), function(value) {
      return /^[a-z\s.]*$/i.test(value); }, "Must use alphabetic latin characters");
    setInputFilter(document.getElementById("state"), function(value) {
      return /^[a-z\s.]*$/i.test(value); }, "Must use alphabetic latin characters");
    setInputFilter(document.getElementById("address"), function(value) {
      return /^[a-z0-9\s.]*$/i.test(value); }, "Must use alphabetic latin characters or numbers");
</script>
```

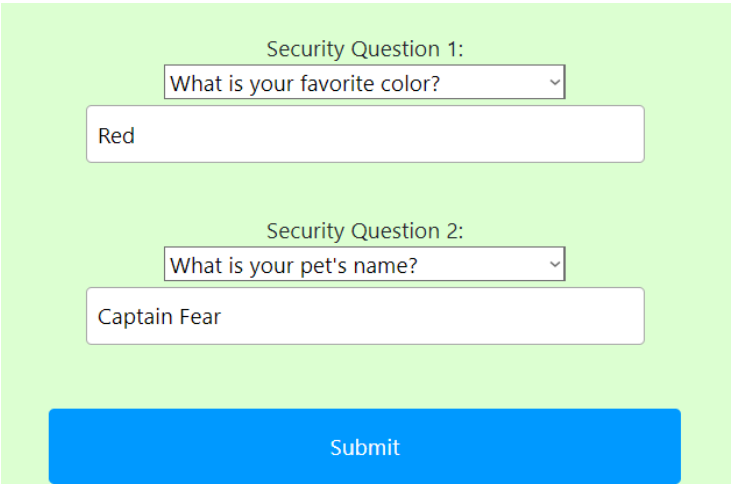
Here is the php code that handles the form submitted data, conducts the SQL queries, and puts the data into both the Addresses and PaymentInfo tables.

```php
<body>
    <?php

        $address = $_REQUEST['address'];
        $city = $_REQUEST['city'];
        $state = $_REQUEST['state'];
        $zipCode = $_REQUEST['zip'];
        $ccNumber = $_REQUEST['crednum'];
        $nameOnCard = $_REQUEST['ccname'];
        $cvv = $_REQUEST['cvv'];


        if($address != '' && $city != '' && $state != '' && $zipCode != '' && $ccNumber != '' && $nameOnCard != '' && $cvv != '') {
            #inserts the address variables from the form into the Addresses table
            $query_addAddress = "INSERT INTO Addresses (address, city, state, zipCode) VALUES ('$address', '$city', '$state', '$zipCo
            mysqli_query($conn, $query_addAddress);

            #find the ID of the last address that was entered
            $query_addressID_select = "SELECT MAX(addressID) FROM Addresses;";
            $result = mysqli_query($conn, $query_addressID_select);
            $resultCheck = mysqli_num_rows($result);

                if ($resultCheck > 0) {
                    while ($row = mysqli_fetch_assoc($result)) {
                        $foundID = $row['MAX(addressID)'];
                    }
                }

            #using the addressID from above, insert the payment info from the form into the PaymentInfo table
            $query_addCC = "INSERT INTO PaymentInfo (ccNumber, nameOnCard, cvv, addressID) VALUES ('$ccNumber', '$nameOnCard', '$cvv'
            mysqli_query($conn, $query_addCC);

            #find the last paymentID that was entered into the table
            $query_paymentID_select = "SELECT MAX(paymentID) FROM PaymentInfo;";
            $result = mysqli_query($conn, $query_paymentID_select);
            $resultCheck = mysqli_num_rows($result);

                if ($resultCheck > 0) {
                    while ($row = mysqli_fetch_assoc($result)) {
                        $payment_id = $row['MAX(paymentID)'];
                    }
                }
```

```php
        #Find the userID using the session
        $newID = $_SESSION['userSessID'];

            $sql = "SELECT userID FROM User NATURAL JOIN Sessions WHERE sessionID = '$newID';";
            $result = mysqli_query($conn, $sql);
            $resultCheck = mysqli_num_rows($result);

            if ($resultCheck > 0) {
                while ($row = mysqli_fetch_assoc($result)) {
                    $owner_id = $row['userID'];
                }
            }

        #update the User table with the paymentID for the userID we got from the above query
        $query_for_paymentID_update = "UPDATE User SET paymentID = '$payment_id' WHERE userID = '$owner_id';";
        mysqli_query($conn, $query_for_paymentID_update);

        $conn->close();


        // redirects to the Payment Plan page
        header("Location: payment_plan.html");
    }
    else {
        echo "Please click the back arrow on your browser and fill out all the fields.";
    }

?>
```

Requirement 2: The system shall require the user to set up a security question for account recovery during account creation.
Competition Rate: 100%
Implementer/Tester: Abhi

Here are the security question inputs (clearly displayed on the setup screen for simpler account creation for the users):



Here is the HTML code for the security questions, including the rest of the setup page:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Account Setup</title>
    <script src="signup.js"></script>

        <!-- Bootstrap CSS Needed for Navbar, etc. -->
        <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.
css" rel="stylesheet"
integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65Vohhpuu
COmLASjC" crossorigin="anonymous">

        <!-- Bootstrap Icons -->
        <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-ic
ons.css">

        <!-- Javascript source for Navbar import code-->
        <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></s
cript>

        <!-- References the main.css file for style alterations -->
      <link rel="stylesheet" type="text/css" href="main.css" id="csslink">

        <!-- Makes the title the name of the tab -->
      <title>Account Setup</title>
    <style>
      /* Centering everything except the logo */
      /* JAY'S NOTE: Commented the body tag selector out as it was messing
up the navbar and logo.
                  Removal didn't affect any other element on page. */
      /* body {
      display: flex;
      flex-direction: column;
      align-items: center;
      } */
```

```
      .container {
      margin-top: 50px;
      text-align: center;
      }

      form {
      margin-top: 20px;
      display: flex;
      flex-direction: column;
      align-items: center;
      }

      /* Styling the logo */
      .imgCenter {
      display: block;
      margin-left: auto;
      margin-right: auto;
      max-width: 100%;
      height: auto;
      }

      /* Styling the requirements */
      .requirements {
      color: red;
      display: block;
      }

      .fulfilled {
      color: green;
      display: block;
      }
    </style>

<link rel="stylesheet" href="theme.css" type="text/css" />
<script src="theme.js"></script>

</head>
<!-- On refresh the page goes to the saved theme option -->
  <!-- Navbar section -->
  <nav id="nav-placeholder">
```

```
  </nav>
  <script>
      $.get("navigation.html", function(data){
          $("#nav-placeholder").replaceWith(data);
        });
    </script>
  <!-- End Navbar section -->

  <body onload="currenttheme()">
    <!-- Displays the logo of PLATFORM -->
    <br><div style="text-align: center;">
      <img src="logo.png" alt="logo" class="imgCenter">
    </div>

    <!-- Displays the title of the page and instructions  -->
    <div class="container">
      <h1 class="size"><br>Sign Up Page</h1>
      <p>Please fill in the information below to create an account.</p>
    </div>

    <form method="post" action="setup-db.php" onsubmit="return
validateForm()">
      <label for="first-name">First Name:</label>
      <input type="text" id="first-name" name="first-name"><br>

      <label for="last-name">Last Name:</label>
      <input type="text" id="last-name" name="last-name"><br>

      <label for="username">Email Address:</label>
      <input type="email" id="username" name="username"><br>

      <label for="password">Password:</label>
      <input type="password" id="password" name="password"><br>

      <label for="account-type">Account Type:</label>
      <select id="account-type" name="account-type">
        <option value="1">Moderator</option>
        <option value="2">Advertiser</option>
        <option value="3">Business Owner</option>
      </select><br>
```

```
      <label for="security-question-1">Security Question 1:</label>
      <select id="security-question-1" name="security-question-1"
required>
        <option value="" disabled selected>Select a question</option>
        <option value="1">What is your mother's maiden name?</option>
        <option value="2">What is your favorite color?</option>
        <option value="3">What is your favorite food?</option>
        <option value="4">What is your pet's name?</option>
      </select>
      <input type="text" id="security-answer-1"
name="security-answer-1"><br>

      <label for="security-question-2">Security Question 2:</label>
      <select id="security-question-2" name="security-question-2"
required>
        <option value="" disabled selected>Select a question</option>
        <option value="1">What is your mother's maiden name?</option>
        <option value="2">What is your favorite color?</option>
        <option value="3">What is your favorite food?</option>
        <option value="4">What is your pet's name?</option>
      </select>
      <input type="text" id="security-answer-2"
name="security-answer-2"><br>

      <input type="submit" value="Submit">
    </form>

    <!-- Bootstrap JavaScript needed for some components to function -
Embed as last thing in body of HTML. -->
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle
.min.js"
integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/t
WtIaxVXM" crossorigin="anonymous"></script>

  </body>
</html>
```

Here is the PHP code for the security questions, including the rest of the setup page:

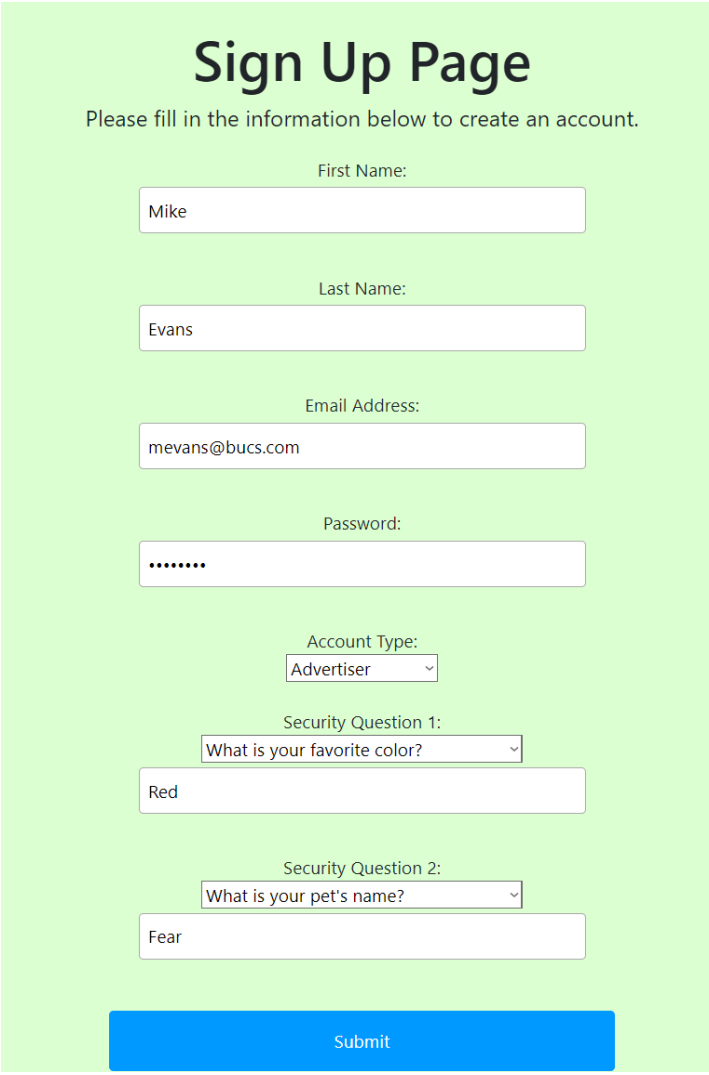
```php
<?php
    ini_set('display_errors', 1);
    ini_set('display_startup_errors', 1);
    error_reporting(E_ALL);


    include "connect.php";
?>

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
</head>
<body>

<?php
    // Check if the form has been submitted
    if (isset($_POST) && !empty($_POST)) {
        // Get form data
        $firstName = $_POST['first-name'];
        $lastName = $_POST['last-name'];
        $email = $_POST['username'];
        $password = $_POST['password'];
        $passwordHash = password_hash($password, PASSWORD_DEFAULT);
        $acctTypeID = intval($_POST['account-type']);
        $secQuesOneID = intval($_POST['security-question-1']);
        $answer1 = $_POST['security-answer-1'];
        $question1Hash = password_hash($answer1, PASSWORD_DEFAULT);
        $secQuesTwoID = intval($_POST['security-question-2']);
        $answer2 = $_POST['security-answer-2'];
        $question2Hash = password_hash($answer2, PASSWORD_DEFAULT);

/*          echo 'Account Type ID: ' . $acctTypeID . '<br>';
        echo 'Security Question 1 ID: ' . $secQuesOneID . '<br>';
        echo 'Security Answer 1: ' . $answer1 . '<br>';
        echo 'Security Answer 1 HASH: ' . $question1Hash . '<br>';
        echo 'Security Question 2 ID: ' . $secQuesTwoID . '<br>';
        echo 'Security Answer 2 : ' . $answer2 . '<br>';
```

```
        echo 'Security Answer 2 HASH : ' . $question2Hash . '<br>';
        exit(); */

        // Insert user data
        $sql = "INSERT INTO User (firstName, lastName, email, acctTypeID,
secQuesOneID, secQuesTwoID)
                VALUES ('$firstName', '$lastName', '$email',
'$acctTypeID', '$secQuesOneID', '$secQuesTwoID')";
        $result1 = mysqli_query($conn, $sql);

        // Get the user ID of the newly inserted user
        $userID = mysqli_insert_id($conn);

        // Insert password hash
        $sql = "INSERT INTO UserPassword (userID, passwordHash) VALUES
('$userID', '$passwordHash');";
        $result2 = mysqli_query($conn, $sql);

        // Insert security question answers
        $sql = "INSERT INTO SecurityQuestionAnswers (userID,
secQuestionID, questionHash)
                VALUES ('$userID', '$secQuesOneID', '$question1Hash'),
                ('$userID', '$secQuesTwoID', '$question2Hash');";
        $result3 = mysqli_query($conn, $sql);

        $conn->close();

        // Redirect to login webpage
        header("Location: login.html");
        exit();
    }
?>
</body>
</html>
```

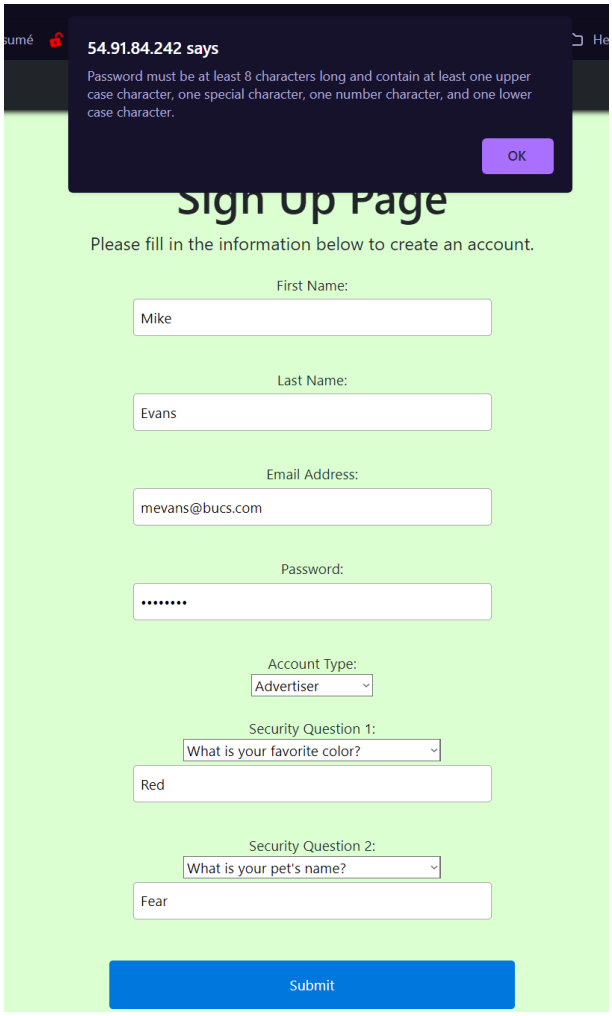
Here are the SecurityQuestions and SecurityQuestionAnswers tables in the database after clicking the Submit button (observe userID = 35):

```
MariaDB [p2]> select * from SecurityQuestions;
+---------------+-----------------------------------+
| secQuestionID | question                          |
+---------------+-----------------------------------+
|             1 | What is your mother's maiden name? |
|             2 | What is your favorite color?      |
|             3 | What is your favorite food?       |
|             4 | What is your pet's name?          |
+---------------+-----------------------------------+
4 rows in set (0.000 sec)

MariaDB [p2]> select * from SecurityQuestionAnswers;
+--------+---------------+--------------------------------------------------------------+
| userID | secQuestionID | questionHash                                                 |
+--------+---------------+--------------------------------------------------------------+
|      6 |             2 | Red                                                          |
|      6 |             4 | Yoshi                                                        |
|      7 |             1 | $2y$10$.nXq2xuoA04BgiifDaSv7uE8kgeKvFMY4GToZHihjFFKF74IGPGzK |
|      7 |             4 | $2y$10$.UvhFNjeh6BUPgC06JPwn.XpvA9.71cC/PMHQOIoi/bZZ92YLU5KW |
|      8 |             1 | $2y$10$rp3DG6c30HPSSl7oeGidauTAG1XtSsU.LBwQxQtdNYKzIDum0C3La |
|      8 |             4 | $2y$10$xrKTh0WsgwI/uaqpvMYHh.ZvRCP5KIvg5.4uH5M/0AL9HvcDVQhUu |
|     17 |             2 |                                                              |
|     17 |             4 |                                                              |
|     20 |             1 |                                                              |
|     20 |             3 |                                                              |
|     22 |             1 |                                                              |
|     22 |             4 |                                                              |
|     24 |             2 |                                                              |
|     24 |             4 |                                                              |
|     25 |             1 |                                                              |
|     25 |             3 |                                                              |
|     26 |             2 | $2y$10$tl74DGP.2b79/UKe5kBe3eP/Ia5mrycqPlH4i26StjkpmA8GGNGjC |
|     26 |             3 | $2y$10$X1JrwPuV4pReCgbEPcuPLefbmyOGvDeEVC3gEWcyO6AnYn47M16Xe |
|     27 |             2 | $2y$10$bcpDMxReqKsMi8g2Zpd0tuhOGya0LiyhuD/1Cq7iPOM1nbJSKUg5m |
|     27 |             4 | $2y$10$KedQ6cjxu9nHNP2.5nzzFunonMCjuYvHwBbx3PT.NpZjPbdRsM52K |
|     28 |             1 | $2y$10$otKgg06geW3A5vWXfVi5zO4ueH.aSs3O859cpMaTbSSdOaAu.7YBC |
|     28 |             2 | $2y$10$SZg0XV7QZGys.v9dQXcJGOvnj47tuYHY6twcZsVlUNj9xlwq.1gAm |
|     29 |             1 | $2y$10$mQgyqrvZBZAWtdHFx.7xNO7OJk4sBuql742m1Ju5JQTRzaqKGTZNC |
|     29 |             2 | $2y$10$SeuBb0N22MclvR5pSHOWLOvwx3OMn0XnleTM4SfHaQcWtrz9fpwCe |
|     30 |             1 | $2y$10$pmR.Fz03wgryJOtcFjyQRO7qWWjApv8gN3pdiwYo5HPV98kaUMAhO |
|     30 |             2 | $2y$10$.IJXrsvmdCUH7yXS6EbG9.wj4h5XvSx9WjGc0SWvOI.wcxKh1gZ4G |
|     31 |             1 | $2y$10$utuzljjKNg9I0MzoM2HwX.QZI1OCrxQVZk7.sKSz6wwOItBxGKw3u |
|     31 |             4 | $2y$10$ncH4NqTiLx3fYm5Emhx33eqZaj9BlLHrd..g7pH2.Jzi9wM86hS36 |
|     32 |             2 | $2y$10$krwy1JXzBAw9.KuORtWM4eiS1eq3wRR705GJnChNt3PF4RIp87E0y |
|     32 |             3 | $2y$10$XopPv3rKwKoqHBlh5NPdOehdELcLDEQvU7sUaI1poQ0GbPIfOmBiu |
|     33 |             1 | $2y$10$WuwgZFQybQT57ZigzGmGnuCIHwo56QJuhWB.LBsvcUJzMEhF4ihHi |
|     33 |             2 | $2y$10$wp7aRtsxFB6QOTTyv5LC8e2AfQQQ2u1k6CG.7RnnL0zd4CN1YrdTe |
|     34 |             1 | $2y$10$gD/ZWtlR0tTy3EQpuWLiH.AEvBHehxF/MuuVo7G1hNU3du/WURjam |
|     34 |             2 | $2y$10$4Z5.wx/rm/2OhKlXOuP4BeDVeoAWNPpTqfPwuS1hyP4dXL2VT8HC2 |
|     35 |             2 | $2y$10$i1QvbOyyVT9X/3xHpZJoxOCvtRz07kdeBi0Q.fHwa0OH9EsBOJ1xO |
|     35 |             4 | $2y$10$tAejOz6/yu4BjpeIZf1Um.zfpZhBXrB0lAfNJ.DJ/up6KCTL9d9F. |
+--------+---------------+--------------------------------------------------------------+
36 rows in set (0.000 sec)
```

Requirement 3: The user shall create a password that is at least 8 characters long using a mix of upper, lower case and at least 1 number and special character.
Competition Rate: 100%
Implementer/Tester: Abhi
Here is the setup page using a valid email (mevans@bucs.com) and invalid password (football):



Here is the error that appears when the password does not fit the requirements:



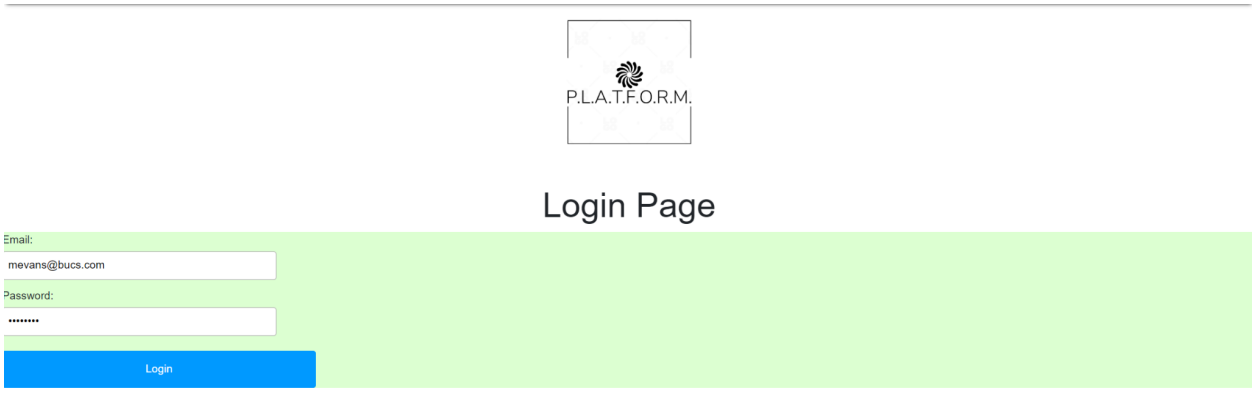
Here is the JavaScript for the setup page (that validates the username and password):

```
function validateForm() {
  var firstName = document.getElementById("first-name").value;
  var lastName = document.getElementById("last-name").value;
  var username = document.getElementById("username").value;
  var password = document.getElementById("password").value;
  var accountType = document.getElementById("account-type").value;
  var question1 = document.getElementById("security-question-1").value;
  var question2 = document.getElementById("security-question-2").value;

  // Check that all fields are filled out
  if (firstName == "" || lastName == "" || username == "" || password ==
"" || accountType == "" || question1 == "" || question2 == "") {
    alert("Please fill out all fields.");
    return false;
```

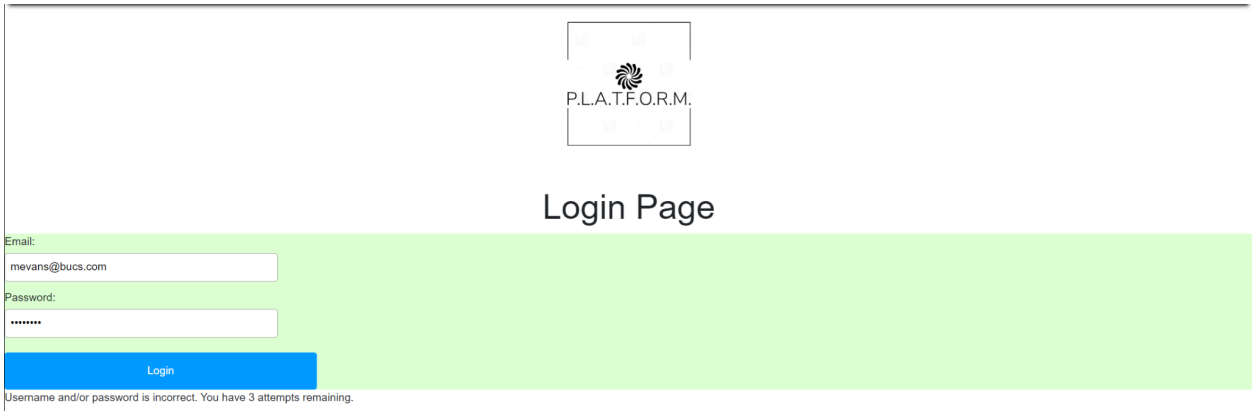
```
  }

  // Check that the username is in email format
  var emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
  if (!emailRegex.test(username)) {
    alert("Please enter a valid email address.");
    return false;
  }

  // Check that the password meets the requirements
  var passwordRegex =
/^(?=.*\d)(?=.*[!@#$%^&*])(?=.*[a-z])(?=.*[A-Z]).{8,}$/;
  if (!passwordRegex.test(password)) {
    alert("Password must be at least 8 characters long and contain at
least one upper case character, one special character, one number
character, and one lower case character.");
    return false;
  }

  // If all checks pass, return true to submit the form
  return true;
}
```

Requirement 4: The system shall lock user accounts after 4 consecutive failed login attempts.
Competition Rate: 100%
Implementer/Tester: Abhi

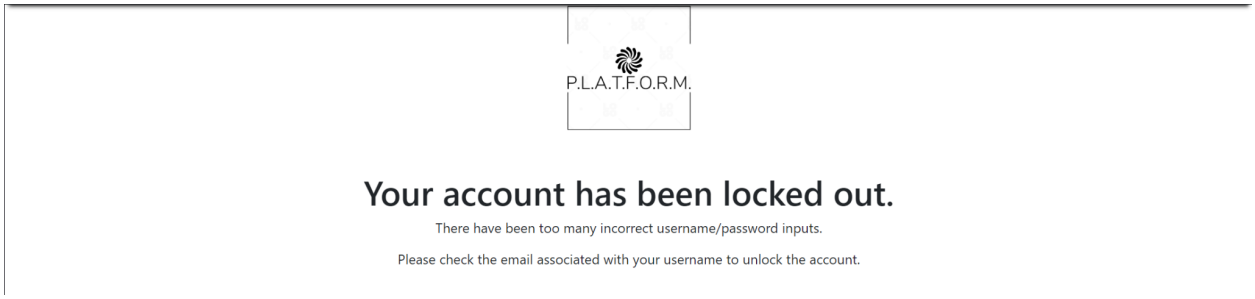
Here is the login page with a valid username (mevans@bucs.com) and an invalid password (football):



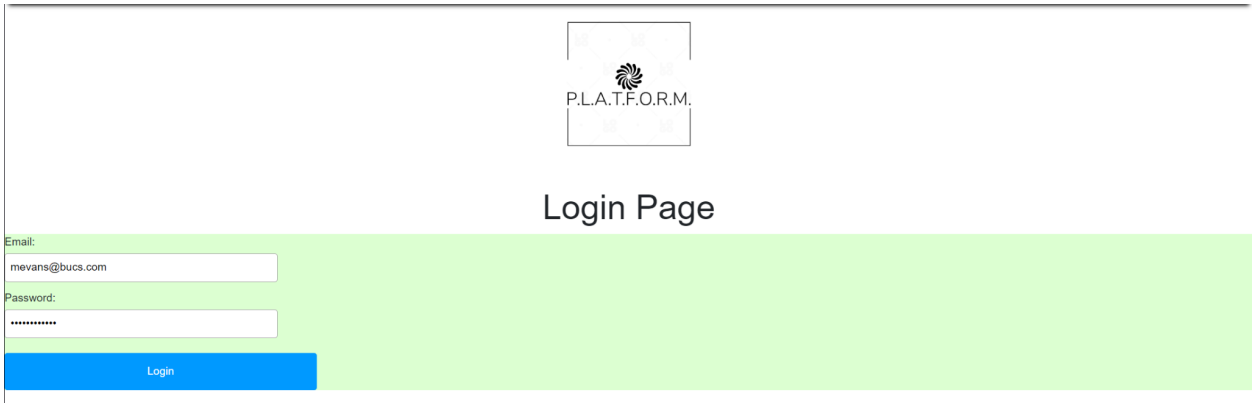
Here is the error that appears after the first wrong password attempt:



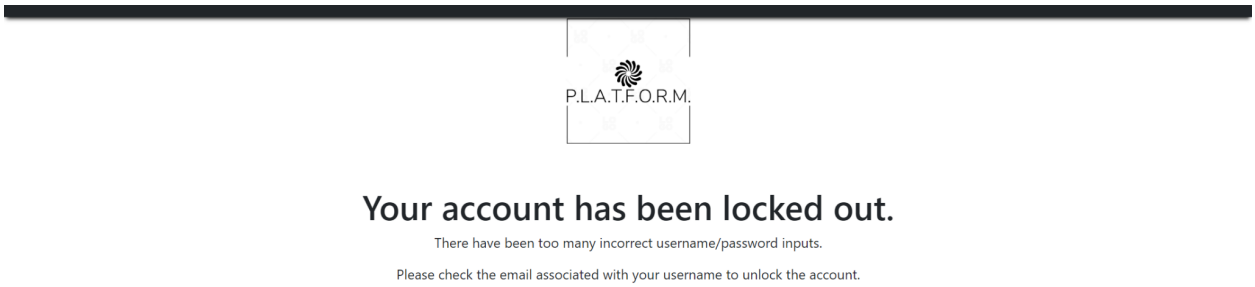
Here is the account lockout screen that appears after 4 incorrect password attempts:



Here is the login page with the valid username and password, after having been locked out earlier:



Here is the lockout screen that appears even if you put in the valid information (the lockout screen will keep on showing up until the user gets their account unlocked, to make sure that only the real user, and not a malicious actor, is logging into the account):



Here is the HTML code for the login page:

```
<!DOCTYPE html>
<html>
  <head>
    <!-- References the main.css file for style alterations -->
    <link rel="stylesheet" type="text/css" href="main.css" id="csslink">

    <!-- Bootstrap CSS Needed for Navbar, etc. -->
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.
css" rel="stylesheet"
integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65Vohhpuu
COmLASjC" crossorigin="anonymous">

    <!-- Bootstrap Icons -->
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-ic
ons.css">

    <!-- Javascript source for Navbar import code-->
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></s
cript>

    <!-- Makes the title the name of the tab -->
    <title>Login Page</title>
    <script src="jquery-3.6.4.js"></script>
    <script src="login.js"></script>

    <style>
```

```
      body {
        font-family: Arial, sans-serif;
      }

      .error-message {
        display: none;
        color: red;
        font-size: 12px;
        margin-bottom: 10px;
      }
    </style>
    <link rel="stylesheet" href="theme.css" type="text/css" />
    <script src="theme.js"></script>

  </head>

<body onload="currenttheme()">
  <!-- Navbar section -->
  <nav id="nav-placeholder">

  </nav>

  <script>
    $.get("navigation.html", function(data){
      $("#nav-placeholder").replaceWith(data);
    });
  </script>
  <!-- End Navbar section -->

  <!-- Displays the logo of PLATFORM and the title of the page -->
  <br><div style="text-align: center;">
    <img src="logo.png" alt="logo" class="imgCenter">
    <h1 class="size"><br>Login Page</h1>
  </div>

    <form id="loginForm" method="POST">
      <label>Email:</label><br>
      <input type="email" name="email" id="email" required><br>

      <label>Password:</label><br>
```

```
      <input type="password" name="password" id="password" required><br>

      <input type="submit" value="Login">
    </form>

    <div id="error"></div>

   <!-- Bootstrap JavaScript needed for some components to function -
Embed as last thing in body of HTML. -->
   <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle
.min.js"
integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/t
WtIaxVXM" crossorigin="anonymous"></script>
  </body>
</html>
```

Here is the PHP code for the login page:

```
<?php
    include "connect.php";
    ini_set('display_errors', 1);
    ini_set('display_startup_errors', 1);
    error_reporting(E_ALL);
?>

<?php
    #This displays errors in the page
    error_reporting(E_ALL);
    ini_set('display_errors', 1);
?>

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
</head>
<body>

<?php
```

```
    var_dump($_POST); // add this line to check the values of $_POST
    $email = $_POST["email"];
    $password = $_POST["password"];

    $sql = "SELECT User.email, UserPassword.passwordHash FROM User
    JOIN UserPassword ON User.userID = UserPassword.userID
    WHERE User.email='$email';";

    $result = mysqli_query($conn, $sql);
    $rows = mysqli_num_rows($result);

    if ($rows > 0) {
        $row = mysqli_fetch_assoc($result);

        if (password_verify($password, $row["passwordHash"])) {
            echo "success";
        } else {
            echo "failure";
        }

//Joaquin's Code for phpsessid
stuff_____________________________________________________________________
_
        #this creates the new phpsessid and starts the session
        $newID = session_create_id();
        session_commit();
        ini_set('session.use_strict_mode', 0);
        session_id($newID);
        session_start();

        #use the email returned from above to find the userID of this
account
        $query_for_userID_select = "SELECT userID FROM User WHERE email =
'$email';";
        $result = mysqli_query($conn, $query_for_userID_select);
        $resultCheck = mysqli_num_rows($result);

        if ($resultCheck > 0) {
            while ($row = mysqli_fetch_assoc($result)) {
                $user_ID = $row['userID'];
```

```
            }
        }

        //check to see if the userID is already in the Sessions table, if
they are, remove it, since we're about to add a new one
        //this should ensure that the same user doesn't have multiple
sessionIDs stored in the Sessions table
        $query_for_check_userID_select = "SELECT userID FROM Sessions
WHERE userID = '$user_ID';";
        $result = mysqli_query($conn, $query_for_check_userID_select);
        $resultCheck = mysqli_num_rows($result);

        if ($resultCheck > 0) {
            $query_for_userID_delete = "DELETE FROM Sessions WHERE userID
= '$user_ID';";
            $result = mysqli_query($conn, $query_for_userID_delete);
        }

        $timestamp = date("Y-m-d H:i:s");

        #using the userID returned from above, and the timestamp variable,
insert the session into the table Sessions
        $query_for_Sessions_insert = "INSERT INTO Sessions (userID,
sessionID, lastSeen) VALUES ('$user_ID', '$newID', '$timestamp');";
        mysqli_query($conn, $query_for_Sessions_insert);

        #this line of code make the newly created phpsessID available to
other pages through the global $_SESSION variable
        $_SESSION['userSessID'] = $newID;

//End of Joaquin's Code for phpsessid
stuff_____

    } else {
        echo "failure";
    }

    $conn->close();
?>
```

```
</body>
</html>
```

Here is the JavaScript code for the login page:

```
$(document).ready(function() {
    var loginAttempts = 0;
    $("#loginForm").submit(function(event) {
        event.preventDefault();
        var email = $("input[name=email]").val();
        var password = $("input[name=password]").val();
        console.log(email, password); // add this line to check the values
of email and password
        $.ajax({
            type: "POST",
            url: "check-account.php",
            data: {email: email},
            success: function(response) {
                if (response.includes("locked")) {
                    window.location.href = "lockout.html";
                } else {
                    $.ajax({
                        type: "POST",
                        url: "login-db.php",
                        data: {email: email, password: password},
                        success: function(response) {
                            if (response.includes("success")) {
                                window.location.href = "ad_campaign.php";
                            } else {
                                loginAttempts++;
                                if (loginAttempts >= 4) {
                                    $.ajax({
                                        type: "POST",
                                        url: "lock-account.php",
                                        data: {email: email},
                                        success: function(response) {
                                            window.location.href =
"lockout.html";
                                        }
                                    });
```

```
                                } else {
                                    $("#error").text("Username and/or
password is incorrect. You have " + (4 - loginAttempts) + " attempts
remaining.");
                                }
                            }
                        });
                }
            }
        });
    });
});
```

Here is the PHP code that checks if the account associated with the user input username is locked:

```
<?php
    include "connect.php";
    ini_set('display_errors', 1);
    ini_set('display_startup_errors', 1);
    error_reporting(E_ALL);
?>

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
</head>
<body>

<?php
    $email = $_POST["email"];

    $sql = "SELECT email, acctLocked FROM User
            WHERE email='$email' AND acctLocked=1;";

    $result = (mysqli_query($conn, $sql));
    $rows = mysqli_num_rows($result);
```

```
    if ($rows > 0) {
        echo "locked";
    } else {
        echo "in";
    }

    $conn->close();
?>

</body>
</html>
```

Here is the PHP code that locks the user's account if there are 4 failed login attempts:

```
<?php
    include "connect.php";
    ini_set('display_errors', 1);
    ini_set('display_startup_errors', 1);
    error_reporting(E_ALL);
?>

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
</head>
<body>

<?php
    $email = $_POST["email"];

    $sql = "UPDATE User SET acctLocked=1 WHERE email='$email';";
    $result = (mysqli_query($conn, $sql));

    if ($result) {
        echo "Success";
    } else {
        echo "Error";
    }
```

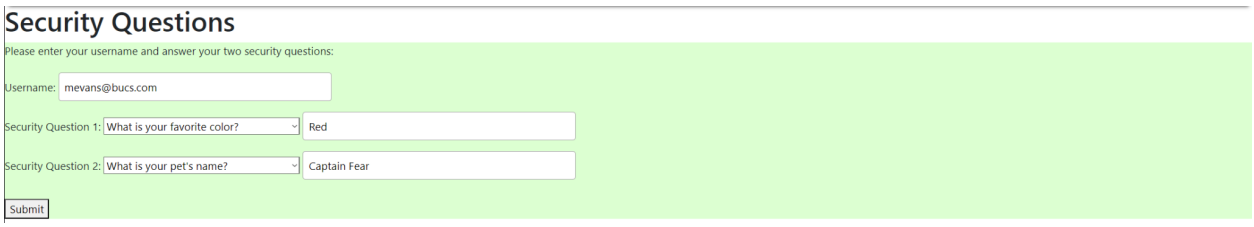
```
    $conn->close();
?>


</body>
</html>
```

Here is the User tables in the database after getting to the locked screen (observe the change in the acctLocked column for userID = 35):

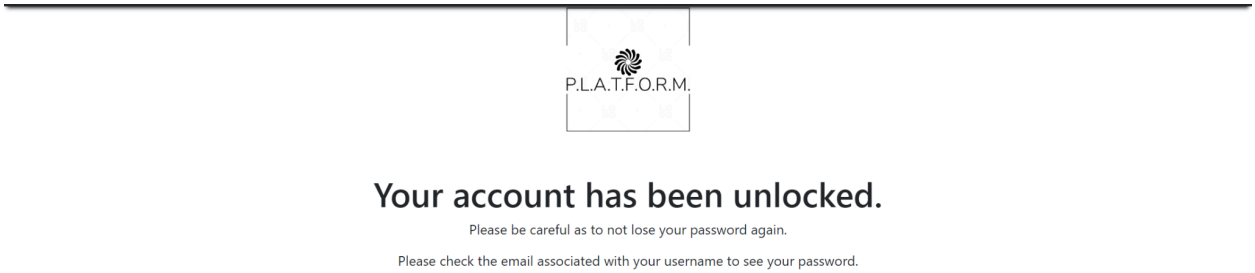
```
MariaDB [p2]> select * from User;
+--------+-----------+----------+--------------------+------------+-------------+-------------+-----------+--------+---------------+---------+------------+
| userID | firstName | lastName | email              | acctTypeID | secQuesOneID | secQuesTwoID | paymentID | planID | numOfAdsOnAcct | newUser | acctLocked |
+--------+-----------+----------+--------------------+------------+-------------+-------------+-----------+--------+---------------+---------+------------+
|      1 | Joaquin   | Merida   | jmerida@usf.edu    |          1 |           2 |           3 |      NULL |      3 |             9 |       1 |          0 |
|      2 | John      | Liegl    | jliegl@usf.edu     |          1 |           3 |           4 |      NULL |      1 |             0 |       1 |          0 |
|      3 | Abhiram   | Nelluri  | anelluri@usf.edu   |          1 |           1 |           2 |      NULL |      0 |             0 |       1 |          0 |
|      4 | Bob       | Bob      | bob@bob.com        |          3 |           1 |           4 |      NULL |      0 |             0 |       1 |          0 |
|      5 | Marley    | Marley   | Marley@bob.com     |          1 |           1 |           3 |      NULL |      0 |             0 |       1 |          0 |
|      6 | Mario     | Mario    | mario@bob.com      |          3 |           2 |           4 |      NULL |      0 |             0 |       1 |          1 |
|      7 | FinalTest | FinalTest | FinTest@bob.com   |          2 |           1 |           4 |      NULL |      0 |             0 |       1 |          1 |
|      8 | Abraham   | Lincoln  | lincoln@bob.com    |          3 |           4 |           1 |      NULL |      0 |             0 |       1 |          0 |
|     17 | Tom       | Brady    | tBrady@gmail.com   |          2 |           2 |           4 |      NULL |      0 |             0 |       1 |          1 |
|     20 | Randy     | Moss     | rMoss@gmail.com    |          3 |           1 |           3 |      NULL |      0 |             0 |       1 |          0 |
|     22 | Tyreek    | Hill     | tHill@gmail.com    |          1 |           1 |           4 |      NULL |      0 |             0 |       1 |          0 |
|     24 | Peter     | Merida   | pete@usf.edu       |          1 |           2 |           4 |        12 |      1 |             0 |       1 |          0 |
|     25 | Panda     | Phantom  | broadsIn@tl.edu    |          1 |           1 |           3 |      NULL |      0 |             0 |       1 |          0 |
|     26 | Drew      | Brees    | brees@saints.com   |          3 |           2 |           3 |      NULL |      0 |             0 |       1 |          0 |
|     27 | Guy       | Fieiri   | flavor@town.com    |          2 |           2 |           4 |      NULL |      0 |             7 |       1 |          0 |
|     28 | Bobby     | Smith    | email@gmail.com    |          2 |           1 |           2 |      NULL |      0 |             0 |       1 |          0 |
|     29 | Yaz       | Yaz      | Yaz@yaz.com        |          2 |           1 |           2 |        17 |      2 |             0 |       1 |          0 |
|     30 | Walter    | White    | email@gmail.com    |          2 |           1 |           2 |      NULL |      3 |             0 |       1 |          0 |
|     31 | SpongeBob | Square   | krusty@gmail.com   |          2 |           1 |           4 |      NULL |      3 |             0 |       1 |          0 |
|     32 | Eugene    | Krabs    | krabs@gmail.com    |          2 |           2 |           3 |        14 |      3 |             0 |       1 |          0 |
|     33 | Clark     | Kent     | superman@gmail.com |          2 |           1 |           2 |        15 |      3 |             0 |       1 |          0 |
|     34 | Bruce     | Wayne    | batman@gmail.com   |          1 |           1 |           2 |        16 |      3 |             1 |       1 |          0 |
|     35 | Mike      | Evans    | mevans@bucs.com    |          2 |           2 |           4 |      NULL |      0 |             0 |       1 |          1 |
+--------+-----------+----------+--------------------+------------+-------------+-------------+-----------+--------+---------------+---------+------------+
23 rows in set (0.000 sec)
```

Requirement 5: The web UI shall provide users with a “forgot password” option to reset their password via email.
(NOTE: The above requirement has been changed slightly given the demo and prototype environment’s inability to configure an SMTP server in which we are working. The new requirement 5 is as follows**: *The web UI shall provide users with a “forgot password” option to unlock their account given accurate security question answers*.)**
Competition Rate: 100%
Implementer/Tester: Abhi

Here is the security questions page:



Here is the recovery page that the user is redirected to after clicking the submit on the security questions page:



Here is the HTML page for the security questions page:

```
<!DOCTYPE html>
<html>
  <head>
  <!-- References the main.css file for style alterations -->
  <link rel="stylesheet" type="text/css" href="main.css" id="csslink">

  <!-- Bootstrap CSS Needed for Navbar, etc. -->
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.
css" rel="stylesheet"
integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65Vohhpuu
COmLASjC" crossorigin="anonymous">

  <!-- Bootstrap Icons -->
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-ic
ons.css">

  <!-- Javascript source for Navbar import code-->
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></s
cript>
  <link rel="stylesheet" href="theme.css" type="text/css" />
  <script src="theme.js"></script>

  <title>Security Questions</title>
  </head>
  <body onload="currenttheme()">
```

```
    <!-- Navbar section -->
    <nav id="nav-placeholder">
    </nav>

    <script>
        $.get("navigation.html", function(data){
            $("#nav-placeholder").replaceWith(data);
        });
    </script>
    <!-- End Navbar section -->

    <h1>Security Questions</h1>
    <form action="security-questions-db.php" method="post">
      <p>Please enter your username and answer your two security
questions:</p>
      <div>
        <label for="username">Username:</label>
        <input type="email" id="username" name="username" required>
      </div>
      <div>
        <label for="security-question-1">Security Question 1:</label>
        <select id="security-question-1" name="security-question-1">
          <option value="1">What is your mother's maiden name?</option>
          <option value="2">What is your favorite color?</option>
          <option value="3">What is your favorite food?</option>
          <option value="4">What is your pet's name?</option>
        </select>
        <input type="text" id="security-answer-1" name="security-answer-1"
required>
      </div>
      <div>
        <label for="security-question-2">Security Question 2:</label>
        <select id="security-question-2" name="security-question-2">
          <option value="1">What is your mother's maiden name?</option>
          <option value="2">What is your favorite color?</option>
          <option value="3">What is your favorite food?</option>
          <option value="4">What is your pet's name?</option>
        </select>
        <input type="text" id="security-answer-2" name="security-answer-2"
required>
```

```
      </div>
      <p id="error-msg"></p>
      <button type="submit">Submit</button>
    </form>

  <!-- Bootstrap JavaScript needed for some components to function - Embed as last thing in body of HTML. -->
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM" crossorigin="anonymous"></script>

  </body>
</html>
```

Here is the PHP page for the security questions page:

```
<?php
  include "connect.php";
?>

<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
</head>
  <body>
    <?php
      // Check if form has been submitted
      $email = $_POST['username'];
      $secQuesOneID = $_POST["security-question-1"];
      $answer1 = $_POST["security-answer-1"];
      $secQuesTwoID = $_POST["security-question-2"];
      $answer2 = $_POST["security-answer-2"];

      $sql = "SELECT userID FROM User WHERE email='$email';";
      $result = mysqli_query($conn, $sql);
      $rows = mysqli_num_rows($result);
```

```
      if($rows == 0) {
        // Username not found
        mysqli_close($conn);
        echo "<script>document.getElementById('error-msg').innerHTML =
'Username not found.';</script>";
        exit();
      } else {
          $row = mysqli_fetch_assoc($result);
          $userID = $row['userID'];
      }

      // Check if userID is associated with the provided security question
IDs
      $sql = "SELECT questionHash FROM SecurityQuestionAnswers WHERE
userID='$userID' AND secQuestionID='$secQuesOneID';";
      $result = mysqli_query($conn, $sql);

      if(mysqli_num_rows($result) == 0) {
          // First security question ID is incorrect
          mysqli_close($conn);
          echo "<script>document.getElementById('error-msg').innerHTML =
'Security Questions/Answers are incorrect.';</script>";
          exit();
      } else {
          $row = mysqli_fetch_assoc($result);
          $question1Hash = $row['questionHash'];
      }

      $sql = "SELECT questionHash FROM SecurityQuestionAnswers WHERE
userID='$userID' AND secQuestionID='$secQuesTwoID';";
      $result = mysqli_query($conn, $sql);

      if(mysqli_num_rows($result) == 0) {
          // Second security question ID is incorrect
          mysqli_close($conn);
          echo "<script>document.getElementById('error-msg').innerHTML =
'Security Questions/Answers are incorrect.';</script>";
          exit();
      } else {
          $row = mysqli_fetch_assoc($result);
```

```
          $question2Hash = $row['questionHash'];
      }

      // Verify that the provided security question answers match the
hashed answers in the table
      if (password_verify($answer1, $question1Hash) &&
password_verify($answer2, $question2Hash)) {
        // Security questions/answers match, update acctLocked to 0
        $sql = "UPDATE User SET acctLocked=0 WHERE email='$email';";
        $result = mysqli_query($conn, $sql);
        mysqli_close($conn);
        header("Location: recovery.html");
      } else {
        // Security question answers do not match
        mysqli_close($conn);
        echo "<script>document.getElementById('error-msg').innerHTML =
'Security Questions/Answers are incorrect.';</script>";
        exit();
      }

    ?>
  </body>
</html>
```

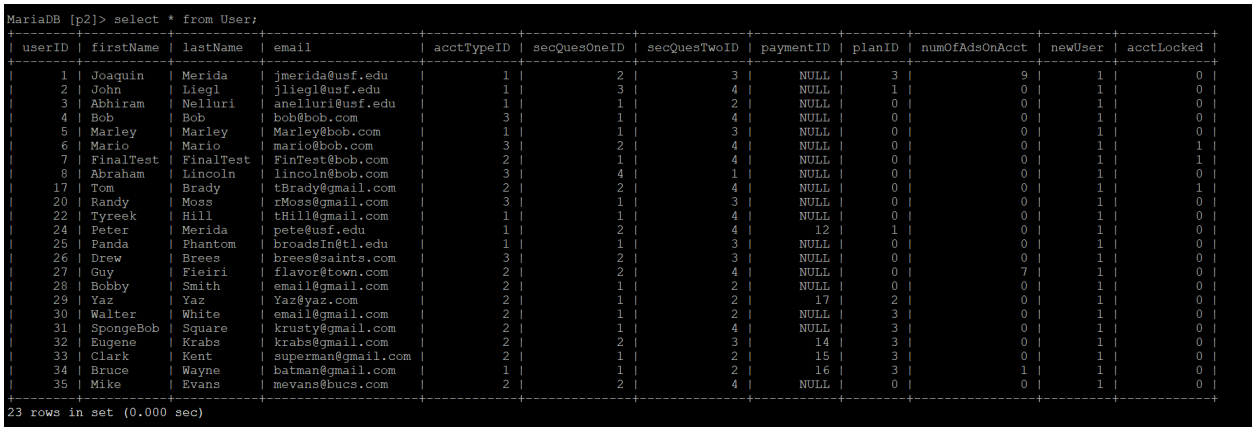
Here is the HTML page for the recovery page:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <!-- References the main.css file for style alterations -->
  <link rel="stylesheet" type="text/css" href="main.css" id="csslink">

  <!-- Bootstrap CSS Needed for Navbar, etc. -->
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.
css" rel="stylesheet"
integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65Vohhpuu
COmLASjC" crossorigin="anonymous">
```

```
  <!-- Bootstrap Icons -->
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-ic
ons.css">

  <!-- Javascript source for Navbar import code-->
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></s
cript>
<link rel="stylesheet" href="theme.css" type="text/css" />
<script src="theme.js"></script>

</head>
<!-- On refresh the page goes to the saved theme option -->
<body onload="currenttheme()">

    <!-- Navbar section -->
    <nav id="nav-placeholder">
    </nav>

    <script>
        $.get("navigation.html", function(data){
            $("#nav-placeholder").replaceWith(data);
        });
    </script>
    <!-- End Navbar section -->

    <!-- Displays the logo for PLATFORM -->
    <div style="text-align: center;">
      <img src="logo.png" alt="logo" class="imgCenter">
    </div>

    <!-- Displays all the text on the page -->
    <div class="container">
      <h1 class="size"><br>Your account has been unlocked.</h1>
      <p>Please be careful as to not lose your password again.</p>
      <p>Please check the email associated with your username to see your
password.</p>
    </div>
```

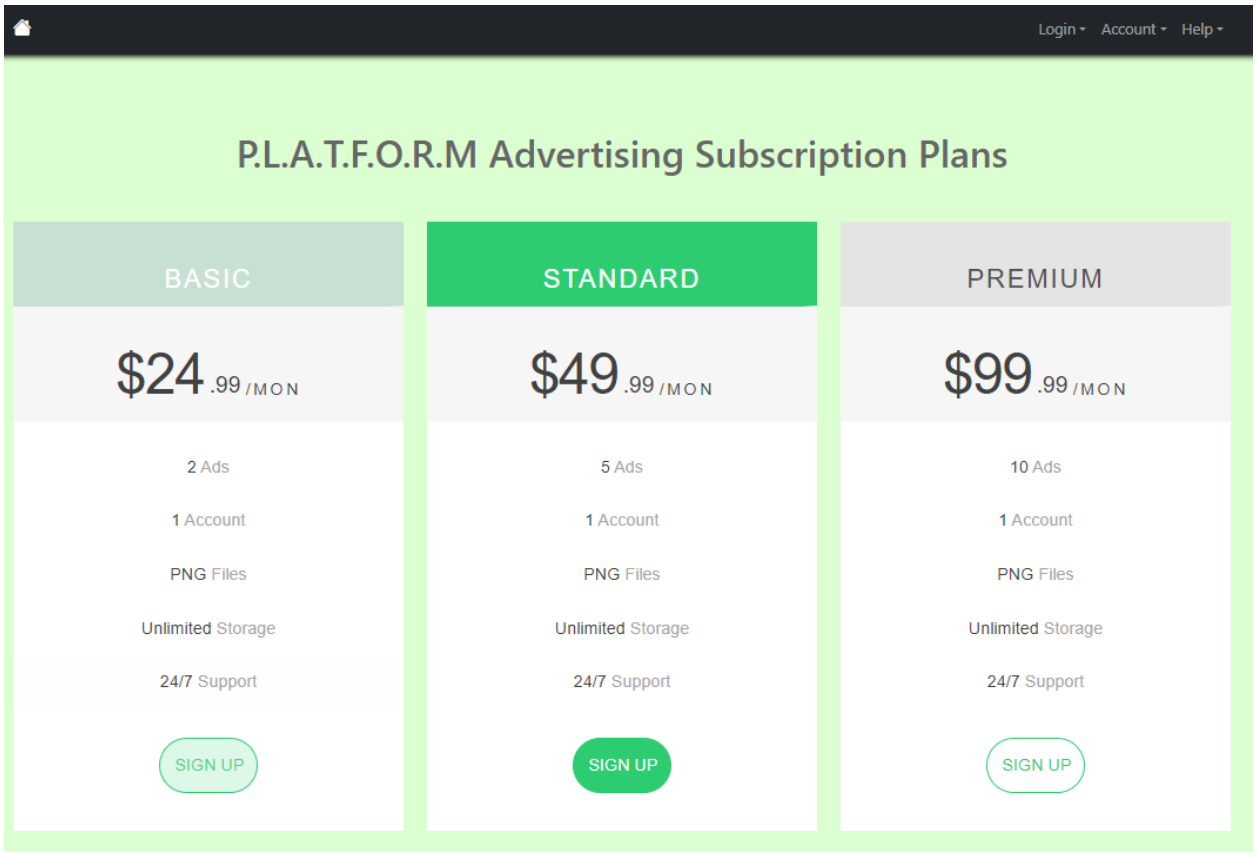
```
  <!-- Bootstrap JavaScript needed for some components to function - Embed as last thing in body of HTML. -->
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM" crossorigin="anonymous"></script>

</body>

</html>
```

Here is the User tables in the database after getting to the unlocked screen (observe the change in the acctLocked column for userID = 35):



```
MariaDB [p2]> select * from User;
+--------+-----------+----------+--------------------+------------+-------------+-------------+-----------+--------+----------------+---------+------------+
| userID | firstName | lastName | email              | acctTypeID | secQuesOneID | secQuesTwoID | paymentID | planID | numOfAdsOnAcct | newUser | acctLocked |
+--------+-----------+----------+--------------------+------------+-------------+-------------+-----------+--------+----------------+---------+------------+
|      1 | Joaquin   | Merida   | jmerida@usf.edu    |          1 |           2 |           3 |      NULL |      3 |              9 |       1 |          0 |
|      2 | John      | Liegl    | jliegl@usf.edu     |          1 |           3 |           4 |      NULL |      1 |              0 |       1 |          0 |
|      3 | Abhiram   | Nelluri  | anelluri@usf.edu   |          1 |           1 |           2 |      NULL |      0 |              0 |       1 |          0 |
|      4 | Bob       | Bob      | bob@bob.com        |          3 |           1 |           4 |      NULL |      0 |              0 |       1 |          0 |
|      5 | Marley    | Marley   | Marley@bob.com     |          1 |           1 |           3 |      NULL |      0 |              0 |       1 |          0 |
|      6 | Mario     | Mario    | mario@bob.com      |          3 |           2 |           4 |      NULL |      0 |              0 |       1 |          1 |
|      7 | FinalTest | FinalTest | FinTest@bob.com   |          2 |           1 |           4 |      NULL |      0 |              0 |       1 |          1 |
|      8 | Abraham   | Lincoln  | lincoln@bob.com    |          3 |           4 |           1 |      NULL |      0 |              0 |       1 |          0 |
|     17 | Tom       | Brady    | tBrady@gmail.com   |          2 |           2 |           4 |      NULL |      0 |              0 |       1 |          1 |
|     20 | Randy     | Moss     | rMoss@gmail.com    |          3 |           1 |           3 |      NULL |      0 |              0 |       1 |          0 |
|     22 | Tyreek    | Hill     | tHill@gmail.com    |          1 |           1 |           4 |      NULL |      0 |              0 |       1 |          0 |
|     24 | Peter     | Merida   | pete@usf.edu       |          1 |           2 |           4 |        12 |      1 |              0 |       1 |          0 |
|     25 | Panda     | Phantom  | broadsIn@tl.edu    |          1 |           1 |           3 |      NULL |      0 |              0 |       1 |          0 |
|     26 | Drew      | Brees    | brees@saints.com   |          3 |           2 |           3 |      NULL |      0 |              0 |       1 |          0 |
|     27 | Guy       | Fieiri   | flavor@town.com    |          2 |           2 |           4 |      NULL |      0 |              7 |       1 |          0 |
|     28 | Bobby     | Smith    | email@gmail.com    |          2 |           1 |           2 |      NULL |      0 |              0 |       1 |          0 |
|     29 | Yaz       | Yaz      | Yaz@yaz.com        |          2 |           1 |           2 |        17 |      2 |              0 |       1 |          0 |
|     30 | Walter    | White    | email@gmail.com    |          2 |           1 |           2 |      NULL |      3 |              0 |       1 |          0 |
|     31 | SpongeBob | Square   | krusty@gmail.com   |          2 |           1 |           4 |      NULL |      3 |              0 |       1 |          0 |
|     32 | Eugene    | Krabs    | krabs@gmail.com    |          2 |           2 |           3 |        14 |      3 |              0 |       1 |          0 |
|     33 | Clark     | Kent     | superman@gmail.com |          2 |           1 |           2 |        15 |      3 |              0 |       1 |          0 |
|     34 | Bruce     | Wayne    | batman@gmail.com   |          1 |           1 |           2 |        16 |      3 |              1 |       1 |          0 |
|     35 | Mike      | Evans    | mevans@bucs.com    |          2 |           2 |           4 |      NULL |      0 |              0 |       1 |          0 |
+--------+-----------+----------+--------------------+------------+-------------+-------------+-----------+--------+----------------+---------+------------+
23 rows in set (0.000 sec)
```

Requirement 6: Users shall be able to pick a payment plan via the web UI, determining duration ads will show on PLATFORM devices.
(NOTE: The above requirement has been changed slightly given the demo and prototype environment in which we are working. The new requirement 6 is as follows**:**
***Users shall be able to pick a payment plan via the web UI, determining the number of ads that they can upload to run on PLATFORM devices*.)**
Competition Rate: 100%
Implementer/Tester: John



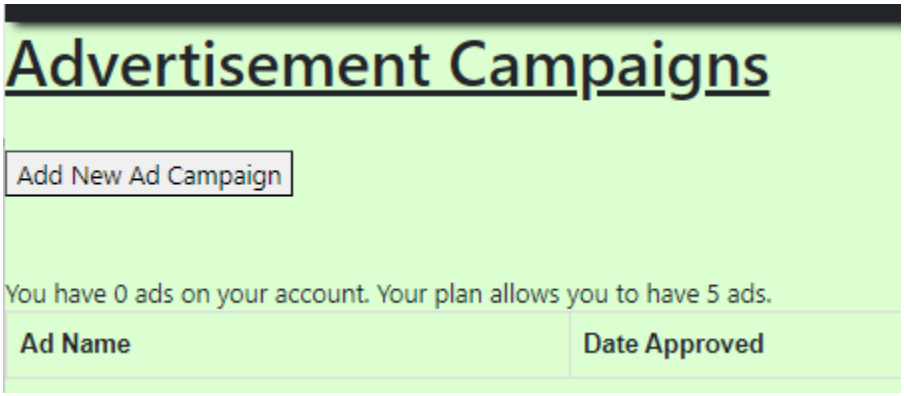
Above we see the available subscription plans. Prior to picking one a new user has a value of 0 for their plan ID in the DB - meaning they have not subscribed yet.

```
MariaDB [p2]> SELECT * FROM User;
+--------+-----------+-----------+--------------------+------------+--------------+--------------+-----------+--------+
| userID | firstName | lastName  | email              | acctTypeID | secQuesOneID | secQuesTwoID | paymentID | planID |
+--------+-----------+-----------+--------------------+------------+--------------+--------------+-----------+--------+
|      1 | Joaquin   | Merida    | jmerida@usf.edu    |          1 |            2 |            3 |      NULL |      3 |
|      2 | John      | Liegl     | jliegl@usf.edu     |          1 |            3 |            4 |      NULL |      1 |
|      3 | Abhiram   | Nelluri   | anelluri@usf.edu   |          1 |            1 |            2 |      NULL |      0 |
|      4 | Bob       | Bob       | bob@bob.com        |          3 |            1 |            4 |      NULL |      0 |
|      5 | Marley    | Marley    | Marley@bob.com     |          1 |            1 |            3 |      NULL |      0 |
|      6 | Mario     | Mario     | mario@bob.com      |          3 |            2 |            4 |      NULL |      0 |
|      7 | FinalTest | FinalTest | FinTest@bob.com    |          2 |            1 |            4 |      NULL |      0 |
|      8 | Abraham   | Lincoln   | lincoln@bob.com    |          3 |            4 |            1 |      NULL |      0 |
|     17 | Tom       | Brady     | tBrady@gmail.com   |          2 |            2 |            4 |      NULL |      0 |
|     20 | Randy     | Moss      | rMoss@gmail.com    |          3 |            1 |            3 |      NULL |      0 |
|     22 | Tyreek    | Hill      | tHill@gmail.com    |          1 |            1 |            4 |      NULL |      0 |
|     24 | Peter     | Merida    | pete@usf.edu       |          1 |            2 |            4 |        12 |      1 |
|     25 | Panda     | Phantom   | broadsIn@tl.edu    |          1 |            1 |            3 |      NULL |      0 |
|     26 | Drew      | Brees     | brees@saints.com   |          3 |            2 |            3 |      NULL |      0 |
|     27 | Guy       | Fieiri    | flavor@town.com    |          2 |            2 |            4 |      NULL |      0 |
|     28 | Bobby     | Smith     | email@gmail.com    |          2 |            1 |            2 |      NULL |      0 |
|     29 | Yaz       | Yaz       | Yaz@yaz.com        |          2 |            1 |            2 |        17 |      0 |
|     30 | Walter    | White     | email@gmail.com    |          2 |            1 |            2 |      NULL |      3 |
```

However, once one of those plans is chosen (Basic = 1, Standard = 2, Premium = 3) this is updated in the User table accordingly, allowing them to upload ads.

```
MariaDB [p2]> SELECT * FROM User;
+--------+-----------+-----------+-------------------+------------+-------------+-------------+-----------+--------+
| userID | firstName | lastName  | email             | acctTypeID | secQuesOneID | secQuesTwoID | paymentID | planID |
+--------+-----------+-----------+-------------------+------------+-------------+-------------+-----------+--------+
|      1 | Joaquin   | Merida    | jmerida@usf.edu   |          1 |           2 |           3 |      NULL |      3 |
|      2 | John      | Liegl     | jliegl@usf.edu    |          1 |           3 |           4 |      NULL |      1 |
|      3 | Abhiram   | Nelluri   | anelluri@usf.edu  |          1 |           1 |           2 |      NULL |      0 |
|      4 | Bob       | Bob       | bob@bob.com       |          3 |           1 |           4 |      NULL |      0 |
|      5 | Marley    | Marley    | Marley@bob.com    |          1 |           1 |           3 |      NULL |      0 |
|      6 | Mario     | Mario     | mario@bob.com     |          3 |           2 |           4 |      NULL |      0 |
|      7 | FinalTest | FinalTest | FinTest@bob.com   |          2 |           1 |           4 |      NULL |      0 |
|      8 | Abraham   | Lincoln   | lincoln@bob.com   |          3 |           4 |           1 |      NULL |      0 |
|     17 | Tom       | Brady     | tBrady@gmail.com  |          2 |           2 |           4 |      NULL |      0 |
|     20 | Randy     | Moss      | rMoss@gmail.com   |          3 |           1 |           3 |      NULL |      0 |
|     22 | Tyreek    | Hill      | tHill@gmail.com   |          1 |           1 |           4 |      NULL |      0 |
|     24 | Peter     | Merida    | pete@usf.edu      |          1 |           2 |           4 |        12 |      1 |
|     25 | Panda     | Phantom   | broadsIn@tl.edu   |          1 |           1 |           3 |      NULL |      0 |
|     26 | Drew      | Brees     | brees@saints.com  |          3 |           2 |           3 |      NULL |      0 |
|     27 | Guy       | Fieiri    | flavor@town.com   |          2 |           2 |           4 |      NULL |      0 |
|     28 | Bobby     | Smith     | email@gmail.com   |          2 |           1 |           2 |      NULL |      0 |
|     29 | Yaz       | Yaz       | Yaz@yaz.com       |          2 |           1 |           2 |        17 |      2 |
|     30 | Walter    | White     | email@gmail.com   |          2 |           1 |           2 |      NULL |      3 |
```

Once a subscription plan has been chosen, the user is prompted with a notification about how many ads they are allowed to have and how many are currently running.



The functionality for taking the user's selection and updating the User table is handled with the following php code on a separate php page. There is one php page for each of the 3 options, the php code for the standard subscription is included below.

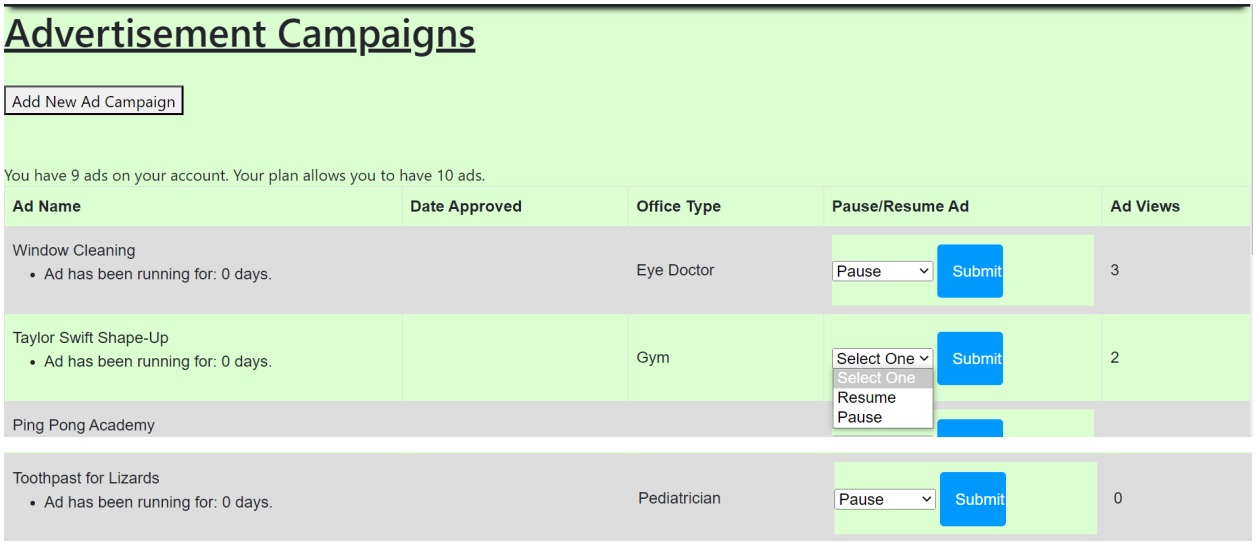
```php
<?php
    #This is a query to find the userID based off the phpsessid
    $newID = $_SESSION['userSessID'];

    $setuserid = "SELECT userID FROM User NATURAL JOIN Sessions WHERE sessionID = '$newID';";
    $result = mysqli_query($conn, $setuserid);
    $resultCheck = mysqli_num_rows($result);

    if ($resultCheck > 0) {
        while ($row = mysqli_fetch_assoc($result)) {
            $user_id = $row['userID'];
        }
    }

    $pmtplan = "UPDATE User SET planID='2' WHERE userID=$user_id";
    $result = mysqli_query($conn, $pmtplan);
    $conn->close();

    // redirects to the Ad Management page
    header("Location: ad_campaign.php");

?>
```

Requirement 7: Users shall be able to suspend/pause recurring membership via the web UI and then restart it later.
Competition Rate: 100%
Implementer/Tester: Joaquin

-When an ad is uploaded, it becomes active by default (value 0 equates to “false” in the adPaused column).

```
MariaDB [p2]> SELECT adID , adOwner , adName , adPaused  FROM Advertisements;
+------+---------+-----------------------+----------+
| adID | adOwner | adName                | adPaused |
+------+---------+-----------------------+----------+
|    1 |       2 | John's Tutoring       |        0 |
|    2 |       1 | Toothpaste for Lizards |        0 |
|    3 |       1 | Window Cleaning       |        0 |
|    4 |       2 | Shoe Shine Street     |        0 |
|    5 |       1 | Taylor Swift Shape-Up |        0 |
|    6 |       1 | Golf Gizmos           |        0 |
|    7 |       1 | Peter Pan Pastries    |        0 |
|    8 |       1 | Abhi's Apple Pies     |        0 |
|    9 |       2 | Raad's Racecars       |        0 |
|   10 |       2 | Joaquin Walkie Talkies |        0 |
|   11 |       1 | Pimp My Ride          |        0 |
|   12 |       2 | Airplane Armadillos   |        0 |
+------+---------+-----------------------+----------+
12 rows in set (0.000 sec)
```

-Each ad has a dropdown option to Pause or Resume the ad. We will pause the *Toothpaste for Lizards* ad.

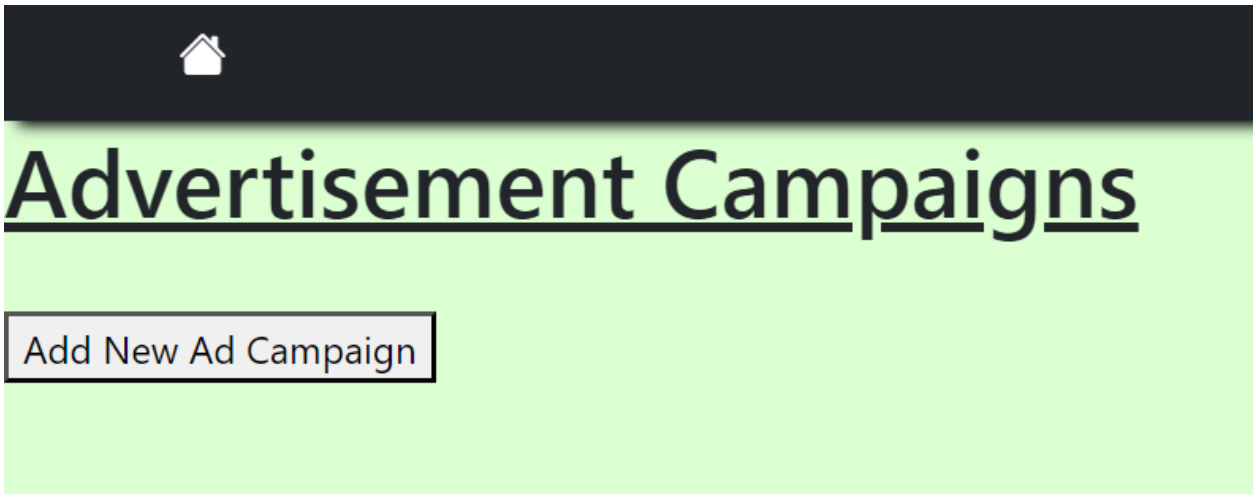


-The change is reflected in our table for Advertisements. As you can see, the adPaused value for Toothpaste for Lizards has been changed to 1, which equates to “true”.

```
MariaDB [p2]> SELECT adID , adOwner , adName , adPaused  FROM Advertisements;
+------+---------+-------------------------+----------+
| adID | adOwner | adName                  | adPaused |
+------+---------+-------------------------+----------+
|    1 |       2 | John's Tutoring         |        0 |
|    2 |       1 | Toothpaste for Lizards  |        1 |
|    3 |       1 | Window Cleaning         |        0 |
|    4 |       2 | Shoe Shine Street       |        0 |
|    5 |       1 | Taylor Swift Shape-Up   |        0 |
|    6 |       1 | Golf Gizmos             |        0 |
|    7 |       1 | Peter Pan Pastries      |        0 |
|    8 |       1 | Abhi's Apple Pies       |        0 |
|    9 |       2 | Raad's Racecars         |        0 |
|   10 |       2 | Joaquin Walkie Talkies  |        0 |
|   11 |       1 | Pimp My Ride            |        0 |
|   12 |       2 | Airplane Armadillos     |        0 |
+------+---------+-------------------------+----------+
12 rows in set (0.000 sec)
```

Requirement 8: The web UI shall allow the user to manipulate and upload an image file/ad to the website, after an account has been created.
Competition Rate: 100%
Implementer/Tester: Joaquin/ Raad(Initial BLOB Implementation/Program Side)

-Once an account has been created, the user will be able to access the Ad Management page which allows them to “Add New Ad Campaign”



-Here is the Advertisements table before the adOwner of ID 2 uploads a new ad.

```
MariaDB [p2]> SELECT adID , adOwner , adName , adPaused  FROM Advertisements WHERE adOwner = 2;
+------+---------+------------------------+----------+
| adID | adOwner | adName                 | adPaused |
+------+---------+------------------------+----------+
|    1 |       2 | John's Tutoring        |        0 |
|    4 |       2 | Shoe Shine Street      |        0 |
|    9 |       2 | Raad's Racecars        |        0 |
|   10 |       2 | Joaquin Walkie Talkies |        0 |
|   12 |       2 | Airplane Armadillos    |        0 |
+------+---------+------------------------+----------+
5 rows in set (0.000 sec)
```

Add New Ad Campaign

What is your userID?: 2

Name of Ad Campaign: Fang Flamethrowers

Image URL: https://lumiere-a.akamaihd.net/v1/images/d-93-incine

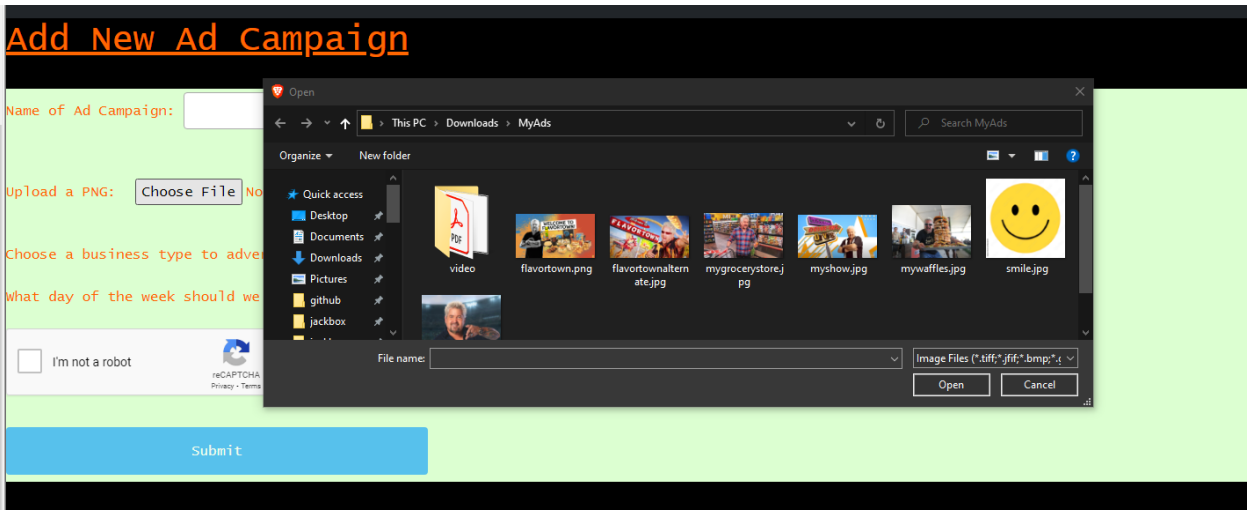
Choose a business type to advertise to: Mechanic Shop

Submit

-The change is reflected in our table for Advertisements.

```
MariaDB [p2]> SELECT adID , adOwner , adName , adPaused  FROM Advertisements WHERE adOwner = 2;
+------+---------+------------------------+----------+
| adID | adOwner | adName                 | adPaused |
+------+---------+------------------------+----------+
|    1 |       2 | John's Tutoring        |        0 |
|    4 |       2 | Shoe Shine Street      |        0 |
|    9 |       2 | Raad's Racecars        |        0 |
|   10 |       2 | Joaquin Walkie Talkies |        0 |
|   12 |       2 | Airplane Armadillos    |        0 |
|   13 |       2 | Fang Flamethrowers     |        0 |
+------+---------+------------------------+----------+
6 rows in set (0.000 sec)
```

The form has an image upload component, that gathers the uploaded binary image data and stores the actual image in the database as BLOB or a Binary Large Object. The image is also gathered, so the PLATFORM program, and can use it to add images to a directory for later display.




```
<label>Upload a PNG:</label>
<input type="file" name="ad_image" accept="image/*">
```

The first variable gathers the image name from the form, The second variable gathers the image binary data

```
$image_name = ($_FILES['ad_image']['name']);
$image_blob = addslashes(file_get_contents($_FILES['ad_image']['tmp_name']));
```

Query to send the info back into the database

```
#This is a query to insert the image data into the Images table
$query_for_Images_insert = "INSERT INTO Images (imageName, image) VALUES ('$image_name', '$image_blob');";
```

The Images table, where this content is stored, the imageid is tied to the advertisements table

imageID	imageName	image
1	john_wick.png	[BLOB]
2	turtle.png	[BLOB]
3	window.png	[BLOB]
4	shoe_shine.png	[BLOB]
5	car.png	[BLOB]
6	tswift.png	[BLOB]
7	golf.png	[BLOB]

On the PLATFORM program, a query is stored that loads up images from the database, this partly looks at the images table and gets the column data of the BLOB column, and the image name column.

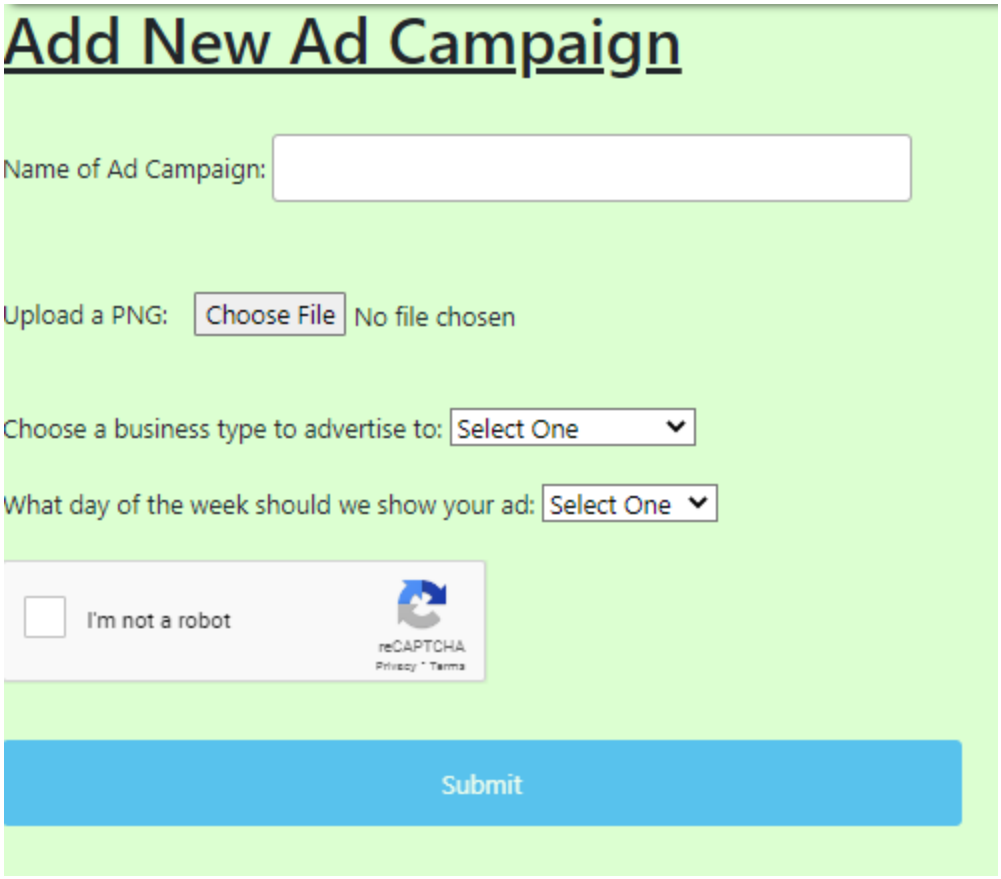
```
full_query1 = query1 + '\'' + where_day + '\''
cursor.execute(full_query1)
data = cursor.fetchall()
for row in data:
    imagedata = row[datacolumnindex]
    namedata = row[namecolumnindex]
    filenames.append(namedata)
    image = Image.open(io.BytesIO(imagedata))
    filepath = filedir + namedata
    rs_image = image.resize((screen_width, screen_height), Image.Resampling.LANCZOS)
    rs_image.save(filepath)
    advertisements.append(ImageTk.PhotoImage(file=filepath))
```

This image blob data is pulled from its column, converted back into a image, resized to fit the screen of the PLATFORM, and save into the directory and new file path the program created.

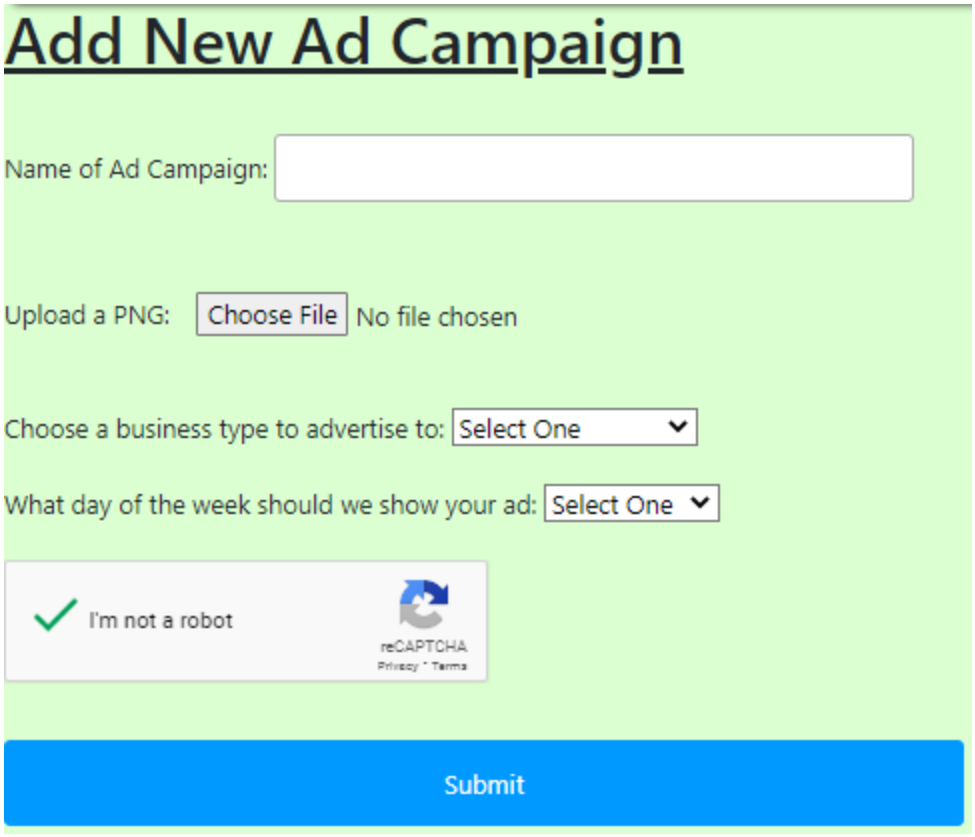
The name data is saved into a file names list, so the program can use it to append the to the directory and create new paths for the images to be stored.
The filenames list is also used for gathering view count statistics from a counter.

Requirement 9: The web UI shall require a secondary confirmation from the user via a reCAPTCHA before any ad is submitted to the system.
Competition Rate: 100%
Implementer/Tester: John

We have implemented a reCAPTCHA on the Add New Ad Campaign page that populates when the page loads. The reCAPTCHA requires the user to click a checkbox in order to be able to submit an ad. Until the checkbox is clicked, the “Submit” button remains grayed out and unresponsive/disabled.



Once the reCAPTCHA checkbox has been clicked and the test has passed, the button becomes active and clickable.

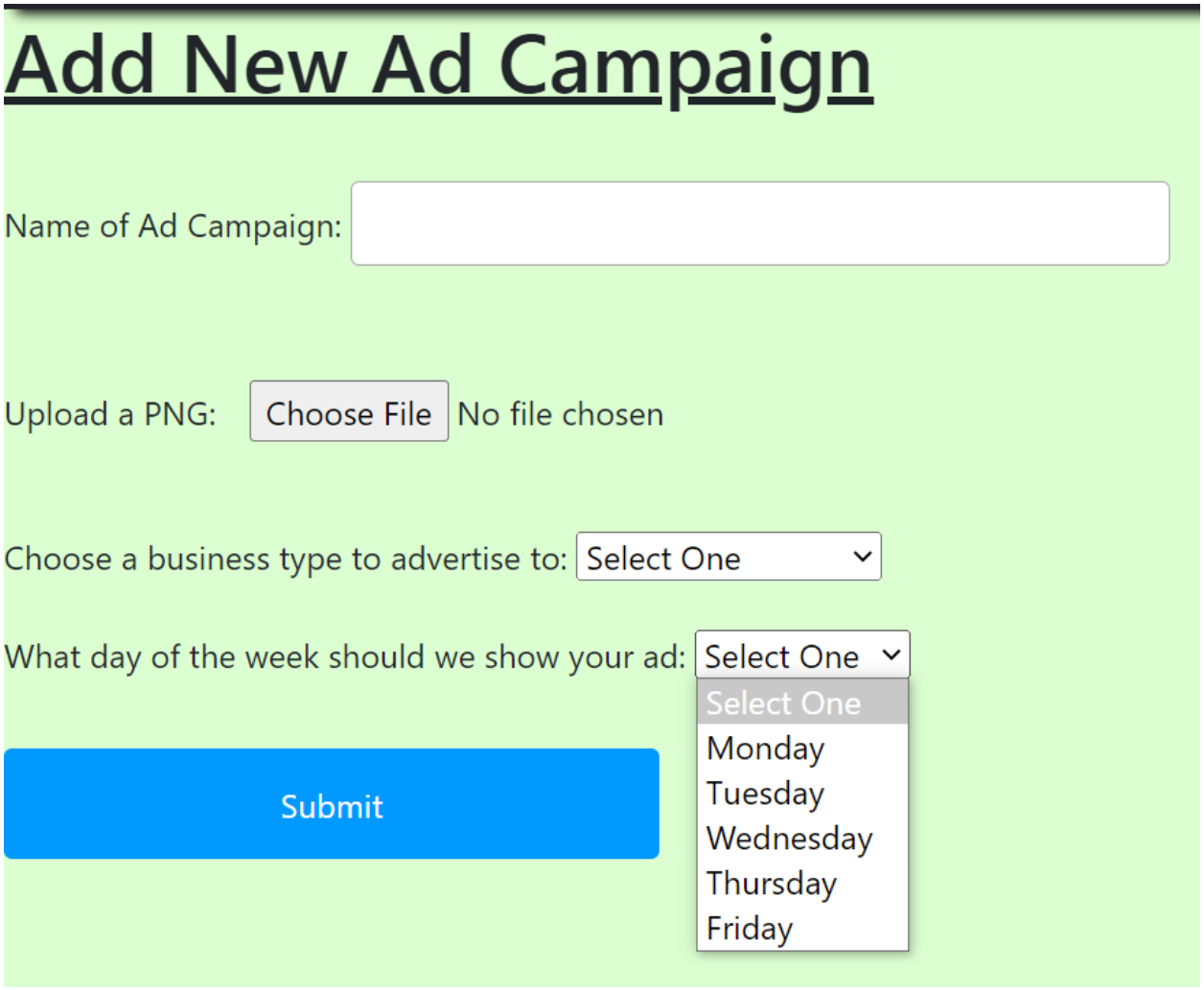


Here is the HTML Code setting up the scripts and HTML body element to display the reCAPTCHA and the associated “Submit” button:

```
<title>Add New Ad Campaign</title>

<!-- code for reCAPTCHA script import -->
<link rel="stylesheet" href="theme.css" type="text/css" />
<script src="theme.js"></script>
<script src="https://www.google.com/recaptcha/api.js" async defer></script>
```

```
    <!-- Create reCAPTCHA -->
    <div
        class="g-recaptcha"
        data-sitekey="6LcWlrAlAAAAAGByXlCLyZlVorfPYUH62bGYxk7l"
        data-callback="callback"
        ></div>
    <br/>

    <input type="submit" id="submit" value="Submit" disabled>

    <!-- <input type="submit" id="submit" value="Submit"> -->

</form>

<!-- Script that reCAPTCHA calls when checkbox clicked to enable button. -->
<script type="text/javascript">
    function callback() {
        const submitButton = document.getElementById("submit");
        submitButton.removeAttribute("disabled");
    }
</script>
```

Requirement 10: The web UI should provide a scheduling mechanism that allows clients to see the availability of advertising time slots.
Competition Rate: 100%
Implementer/Tester: Joaquin / Raad(Program Side)



-The Add New Ad Campaign form allows a user to select which day of the week to display their ads on.The day the user selected is sent back into the database.

At start, the program gets the current day from the datetime.now function. Then there is an if statement that changes a variable to the current day. The day variable is appended into the sql query the program used to extract all images from the database on the where clause. This is the query that is imputed to get the images based on a day. The day variable is appended after the days=.

select * from Images inner join Advertisements on Images.imageID = Advertisements.imageID where adApproved = 1 and adPaused !=1 and days =

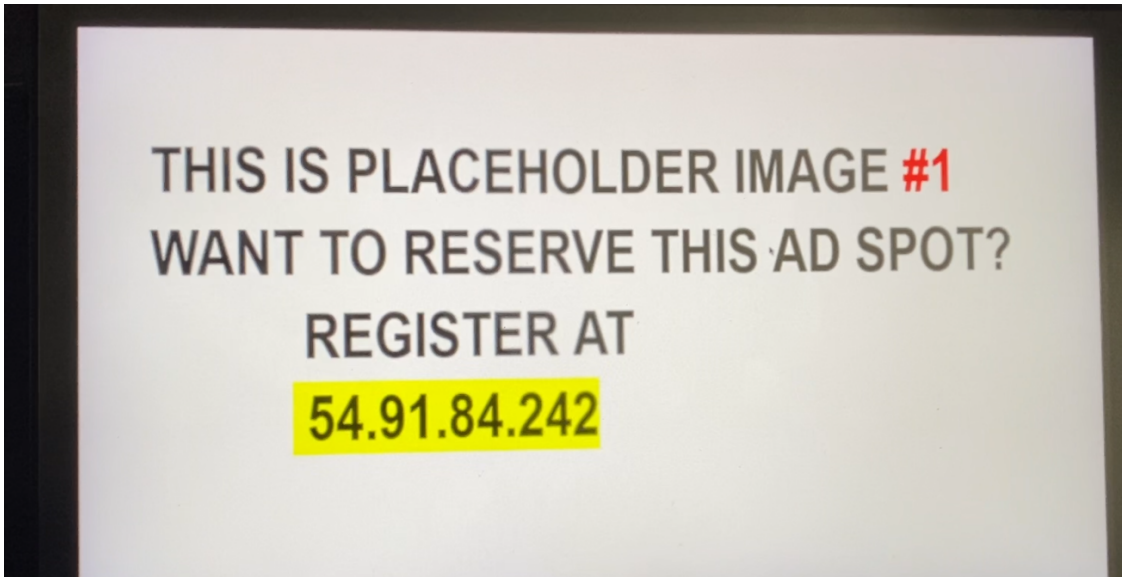
```
def get_day():
    global day
    day = datetime.now()


# This function gets the blobs from database and stores them into the advertisements directory. This function also returns how many advertisements are in the db
def retrieve_image(host1, port1, database1, user1, password1, namecolumnindex, datacolumnindex, plnamecolumnindex, pldatacolumnindex,  query1, query2, query3, filedir):
    global advertisements
    global adcount
    global connection
    global cursor
    global filenames
    advertisements = []
    filenames = []
    connection = mysql.connector.connect(host=host1, port=port1, database=database1, user=user1, password=password1)
    cursor = connection.cursor()
    # This is used for scheduling mechanism, there are certain amount of ads that play per day, we are probably allowing 6 ads to play equally per minute at 10 second inte
    # This part of the program gets the current day, and gets the image ads in the database with the current day
    # This scheduling mechanism is supposed to be on a monthly basis.
    get_day()
    where_day = ""
    if day.strftime('%A') == "Monday":
        where_day = day.strftime('%A')
    if day.strftime('%A') == "Tuesday":
        where_day = day.strftime('%A')
    if day.strftime('%A') == "Wednesday":
        where_day = day.strftime('%A')
    if day.strftime('%A') == "Thursday":
        where_day = day.strftime('%A')
    if day.strftime('%A') == "Friday":
        where_day = day.strftime('%A')
    # Extract the image file name and image that are in the current day
    full_query1 = query1 + '\'' + where_day + '\''
```

For the PLATFORM, we intend to allow 6 ad spots per day, with 10 second intervals between ads.

If 6 approved ads have the current day in the database, the 6 ads displayed normally on the PLATFORM display.

However, there are 6 placeholder images, a placeholder image is added to be displayed if there are less than 6 ads being played.
For each ad not there, a new placeholder image is added to the display

For example if 5 ads are being shown, there would be only one placeholder image.



If there are less than 3, images, there will be placeholder images, with this being the last

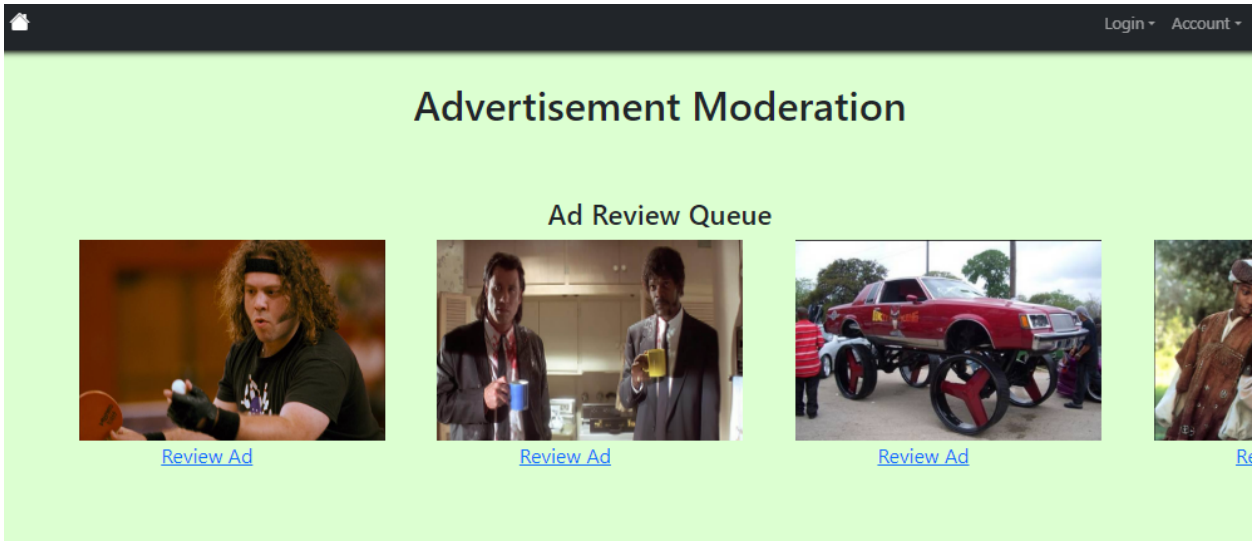
The placeholder image algorithm uses a modified version of the code, that is used to gather the image BLOB data from the database into the advertisement image array

```
if adcount < 6:
    cursor.execute(query3)
    data2 = cursor.fetchall()
    for row2 in data2[:6-adcount]:
            imagedata = row2[pldatacolumnindex]
            namedata = row2[plnamecolumnindex]
            image = Image.open(io.BytesIO(imagedata))
            filepath = filedir + namedata
            rs_image = image.resize((screen_width, screen_height), Image.Resampling.LANCZOS)
            rs_image.save(filepath)
            advertisements.append(ImageTk.PhotoImage(file=filepath))
    adcount=6


if adcount ==0:
    cursor.execute(query3)
    data2 = cursor.fetchall()
    for row2 in data2:
            imagedata = row2[pldatacolumnindex]
            namedata = row2[plnamecolumnindex]
            filenames.append(namedata)
            image = Image.open(io.BytesIO(imagedata))
            filepath = filedir + namedata
            rs_image = image.resize((screen_width, screen_height), Image.Resampling.LANCZOS)
            rs_image.save(filepath)
            advertisements.append(ImageTk.PhotoImage(file=filepath))
    adcount=6
```

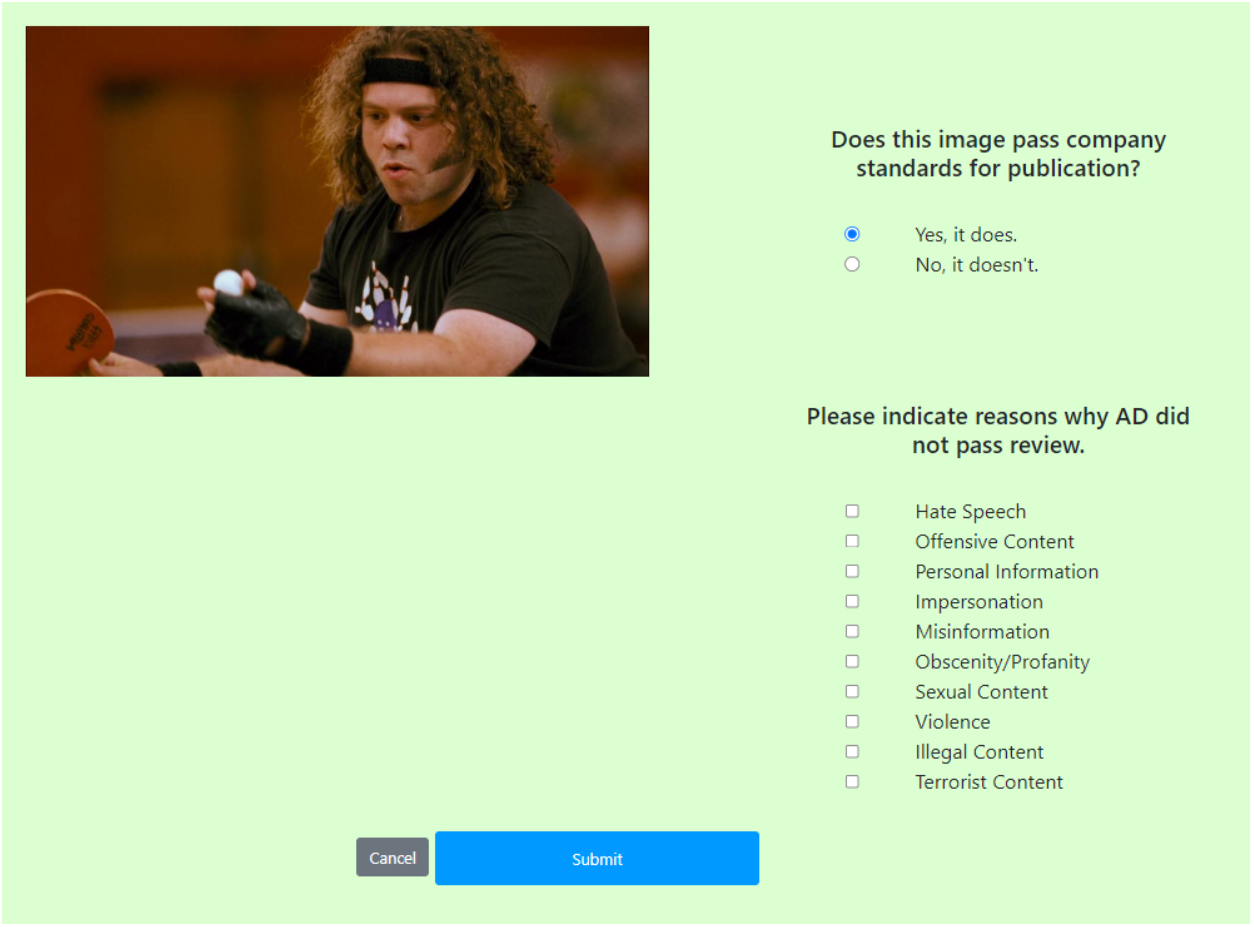
Requirement 11: Moderators will approve ads submitted by users and send them to the appropriate PLATFORM locations as defined by the user.
(NOTE: The above requirement has been changed slightly given the demo and prototype environment in which we are working. The new requirement 11 is as follows**:**
Moderators will approve ads submitted by users enabling PLATFORM devices to retrieve and display them as defined by the user.)
Competition Rate: 100%
Implementer/Tester: John

The moderation page actually consists of two separate pages. The first is the moderator review queue which queries the database, pulls, and displays all ads that are marked with a “0” in the “adApproved” field indicating a “false” status for whether they have been approved or not.

```
MariaDB [p2]> SELECT * FROM Advertisements;
+------+---------+-----------------------------+---------+----------------+----------------+------------+
| adID | adOwner | adName                      | imageID | businessTypeID | displayCounter | adApproved |
+------+---------+-----------------------------+---------+----------------+----------------+------------+
|    1 |       2 | Johns Tutoring              |       1 |              2 |              3 |          1 |
|    2 |       1 | Toothpast for Lizards       |       2 |              5 |              0 |          1 |
|    3 |       1 | Window Cleaning             |       3 |              3 |              3 |          1 |
|    4 |       2 | Shoe Shine Street           |       4 |             10 |              3 |          1 |
|    6 |       1 | Taylor Swift Shape-Up       |       6 |              4 |              2 |          1 |
|    7 |       1 | Golf Gizmos                 |       7 |             10 |              4 |          1 |
|   12 |       1 | Ping Pong Academy           |      13 |              4 |              0 |          0 |
|   20 |       1 | Monday Macchiato            |      21 |              6 |              0 |          0 |
|   23 |      27 | flavor town                 |      26 |              7 |              4 |          1 |
|   25 |      27 | Fieri grocery store         |      28 |             10 |              4 |          1 |
|   26 |      27 | Diners,Drive ins, and Dives! |      29 |              7 |              4 |          1 |
|   29 |      27 | Yummy Waffles?              |      33 |              7 |              4 |          1 |
|   33 |      27 | Fast Car                    |      41 |              2 |              0 |          0 |
|   34 |       1 | Shoe Shine Service          |      51 |             10 |              0 |          0 |
|   35 |       1 | Tasty Treats                |      52 |              5 |              0 |          0 |
|   36 |       1 | Crazy Frog                  |      53 |              4 |              0 |          0 |
|   37 |      27 | Johny Gunfu                 |      54 |              4 |              0 |          0 |
|   38 |      34 | Smile MORE                  |      55 |              7 |              0 |          0 |
|   39 |      27 | super bowl                  |      56 |              7 |              0 |          0 |
+------+---------+-----------------------------+---------+----------------+----------------+------------+
19 rows in set (0.000 sec)
```



We will start with adID # 12 - Ping Pong Academy - which is the first photo on the left. When you click the “Review Ad” link under the photograph it takes you to the Ad Review page where a moderator can determine if the ad passes review or not.



On this page the ad image is pulled in a larger size, and the reviewer has the option to approve the ad or fail the ad for moderation purposes. There is also a set of checkboxes listing the reasons to select indicating any reasons why an ad may not pass review. If the “yes” radio button is picked, passing the ad, and the submit button is clicked, it updates the database, indicating the ad passed review and can be displayed on a PLATFORM device in an ad rotation. Also, while not shown in the database captures, the page also updates the database with a “date reviewed” timestamp which is later utilized for statistical/analytical purposes by other pages. The website also returns the moderator back to the Ad Review Queue to continue the review process for any remaining ads.

```
MariaDB [p2]> SELECT * FROM Advertisements;
+------+---------+------------------------------+---------+----------------+----------------+------------+
| adID | adOwner | adName                       | imageID | businessTypeID | displayCounter | adApproved |
+------+---------+------------------------------+---------+----------------+----------------+------------+
|    1 |       2 | Johns Tutoring               |       1 |              2 |              3 |          1 |
|    2 |       1 | Toothpast for Lizards        |       2 |              5 |              0 |          1 |
|    3 |       1 | Window Cleaning              |       3 |              3 |              3 |          1 |
|    4 |       2 | Shoe Shine Street            |       4 |             10 |              3 |          1 |
|    6 |       1 | Taylor Swift Shape-Up        |       6 |              4 |              2 |          1 |
|    7 |       1 | Golf Gizmos                  |       7 |             10 |              4 |          1 |
|   12 |       1 | Ping Pong Academy            |      13 |              4 |              0 |          1 |
|   20 |       1 | Monday Macchiato             |      21 |              6 |              0 |          0 |
|   23 |      27 | flavor town                  |      26 |              7 |              4 |          1 |
|   25 |      27 | Fieri grocery store          |      28 |             10 |              4 |          1 |
|   26 |      27 | Diners,Drive ins, and Dives! |      29 |              7 |              4 |          1 |
|   29 |      27 | Yummy Waffles?               |      33 |              7 |              4 |          1 |
|   33 |      27 | Fast Car                     |      41 |              2 |              0 |          0 |
|   34 |       1 | Shoe Shine Service           |      51 |             10 |              0 |          0 |
|   35 |       1 | Tasty Treats                 |      52 |              5 |              0 |          0 |
|   36 |       1 | Crazy Frog                   |      53 |              4 |              0 |          0 |
|   37 |      27 | Johny Gunfu                  |      54 |              4 |              0 |          0 |
|   38 |      34 | Smile MORE                   |      55 |              7 |              0 |          0 |
|   39 |      27 | super bowl                   |      56 |              7 |              0 |          0 |
+------+---------+------------------------------+---------+----------------+----------------+------------+
19 rows in set (0.000 sec)
```

You can see that “adApproved” has changed from “0” to “1” meaning that it is “true” that the ad has been approved. Accordingly the Ad has dropped off the Ad Review Queue.

Here is the php code that displays the ads in a row on the Ad Review Queue page and creates the links below them to take the moderator to the approval page.

```php
<?php

    $get_pics = "SELECT * FROM Images NATURAL JOIN Advertisements WHERE adApproved = 0;";
    $result = mysqli_query($conn, $get_pics);
    $resultCheck = mysqli_num_rows($result);

    if ($resultCheck > 0) {
        echo "<table>";
        echo "<tr>";
        while($row = mysqli_fetch_array($result)){

            echo "<td>";
            echo '<img src="'.base64_encode($row['image'] ).'" height="200" width="355"/>';
            ?><a href="ad_approval.php?id=<?php echo $row["imageID"]; ?>">Review Ad</a> <?php
            echo "</td>";

        }
        echo "</tr>";
        echo "</table>";
    }
?>
```

Here is the code for the Ad approval page which takes the adID number forwarded from the last page from a POST statement, saves it to a variable, uses it to pull the photo into the page as an element, and then forwards that same adID number to the “ad_approved_DB.php” handler to update the database accordingly. (NOTE: The commented out “$id” variable was moved to the top of the code so it was accessible in other areas of the page.)

```php
<form action="ad_approved_DB.php?id=<?php echo $id; ?>" method="post" id="ad_form">
    <div class="container"><br><br>
        <div class="row">
            <div class="col-lg-8"> <!-- There are 12 columns, this sets this column to take up 8, and to break and stack columns a

                <?php
                    // $id = $_GET["id"];
                    $get_pic = "SELECT * FROM Images WHERE imageID = $id;";
                    $result = mysqli_query($conn, $get_pic);
                    $resultCheck = mysqli_num_rows($result);

                    if ($resultCheck > 0) {
                        while($row = mysqli_fetch_array($result)) {
                            echo '<img src="'.base64_encode($row['image'] ).'" height="350" width="623"/>';
                        }
                    }
                ?>

            </div>
            <div class="col-lg-4 my-auto"> <!-- my-auto vertically centers content in column -->
                <h4>Does this image pass company standards for publication?</h4>
                <br>
                <div class="d-flex justify-content-start"> <!-- d-flex justify-content-start sets the checkbox/label to justify le
                    <input type="radio" id="pass" name="mod_review" value="true" checked>
                    <label for="pass">Yes, it does.</label>
                </div>
                <div class="d-flex justify-content-start">
                    <input type="radio" id="fail" name="mod_review" value="false">
                    <label for="fail">No, it doesn't.</label>
                </div>
            </div>
        </div>
    </div>
    <br>
```

```
<div class="container">
    <div class="row">
        <div class="col-lg-8">
        <!-- A textbox could go in here, but that is outside the scope of this project at th
        </div>
        <br>
        <div class="col-lg-4"> <!-- This col-lg-* setting causes the columns to stack at a w
            <h4>Please indicate reasons why AD did not pass review.</h4>
            <br>

            <div class="d-flex justify-content-start">
                <input type="checkbox" id="hate_spch" name="reason_failed_rvw">
                <label for="hate_spch">Hate Speech</label>
            </div>
            <div class="d-flex justify-content-start">
                <input type="checkbox" id="offensive" name="reason_failed_rvw">
                <label for="offensive">Offensive Content</label>
            </div>
            <div class="d-flex justify-content-start">
                <input type="checkbox" id="pers_info" name="reason_failed_rvw">
                <label for="pers_info">Personal Information</label>
            </div>
            <div class="d-flex justify-content-start">
                <input type="checkbox" id="Impersonate" name="reason_failed_rvw">
                <label for="Impersonate">Impersonation</label>
            </div>
            <div class="d-flex justify-content-start">
                <input type="checkbox" id="misinfo" name="reason_failed_rvw">
                <label for="misinfo">Misinformation</label>
            </div>
            <div class="d-flex justify-content-start">
                <input type="checkbox" id="obscenity" name="reason_failed_rvw">
                <label for="obscenity">Obscenity/Profanity</label>
            </div>
            <div class="d-flex justify-content-start">
                <input type="checkbox" id="explcit_content" name="reason_failed_rvw">
                <label for="explcit_content">Sexual Content</label>
            </div>
```

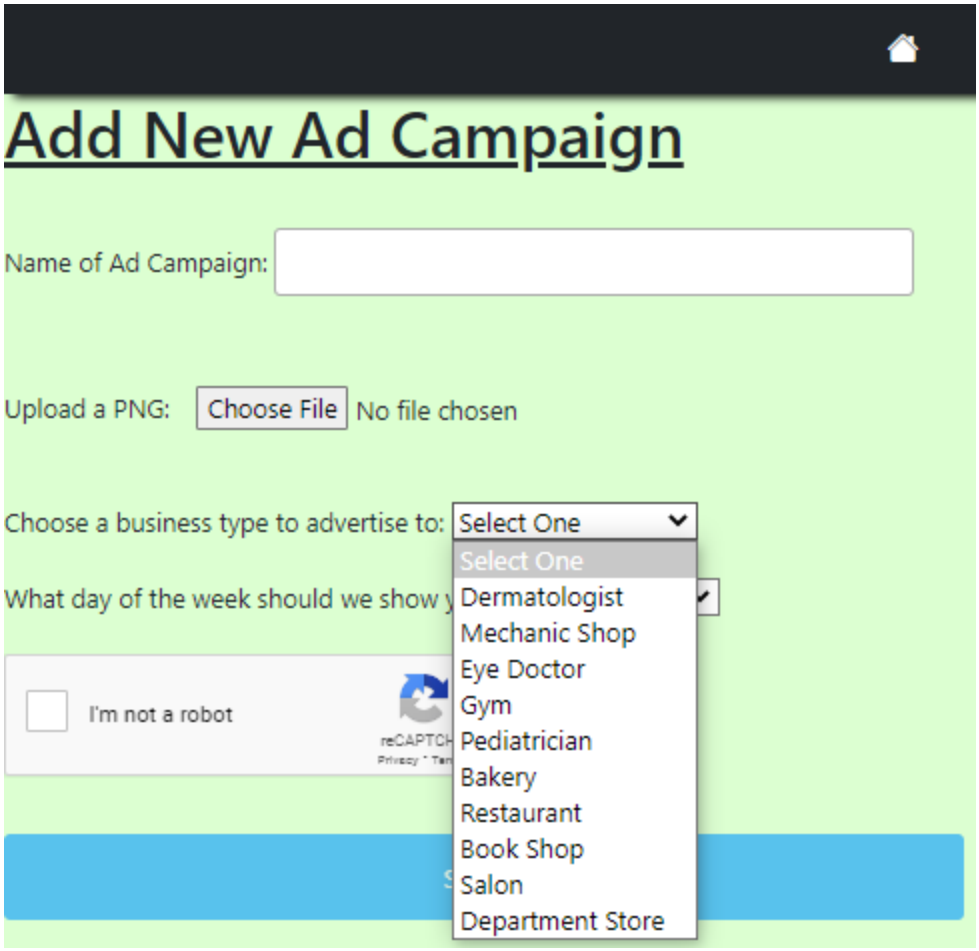
```
                <div class="d-flex justify-content-start">
                    <input type="checkbox" id="illegal" name="reason_failed_rvw">
                    <label for="illegal">Illegal Content</label>
                </div>
                <div class="d-flex justify-content-start">
                    <input type="checkbox" id="terror" name="reason_failed_rvw">
                    <label for="terror">Terrorist Content</label>
                </div>
            </div>
        </div>
    </div>
    <br>
    <div class="container">
        <div class="row">

            <div class="col-md">
                <a class="btn btn-secondary" href="ad_review.php">Cancel</a>
                <input class="btn btn-primary" type="submit" value="Submit">
            </div>
        </div>
    </div>
</form>
```

And here we have the “ad_approved_DB.php” code which takes the forwarded adID information from the database, and uses it and the radio button information to process whether the ad was marked that it passed or failed the ad approval process and update the database accordingly.

```php
<body>
    <?php

        $reviewed = $_REQUEST['mod_review'];
        $id = $_GET["id"];

        if($reviewed == 'true') {

            $update_approved = "UPDATE Advertisements SET adApproved = 1 WHERE imageID = '$id';";
            mysqli_query($conn, $update_approved);

            $update_date_approved = "UPDATE Advertisements SET dateApproved = now() WHERE imageID = '$id';";
            mysqli_query($conn, $update_date_approved);

            $conn->close();
        }
        else {

            $update_date_approved = "UPDATE Advertisements SET dateApproved = now() WHERE imageID = '$id';";
            mysqli_query($conn, $update_date_approved);

            $conn->close();

        }

        // redirects to the Advertisement Moderation page.
        header("Location: ad_review.php");

    ?>
</body>
```

Requirement 12: Users shall be able to make choices via the web UI that allow them to target their ad toward specific medical offices or business types.
Competition Rate: 100%
Implementer/Tester: Joaquin

-The webpage form for adding a new ad lets the user select a specific type of business.



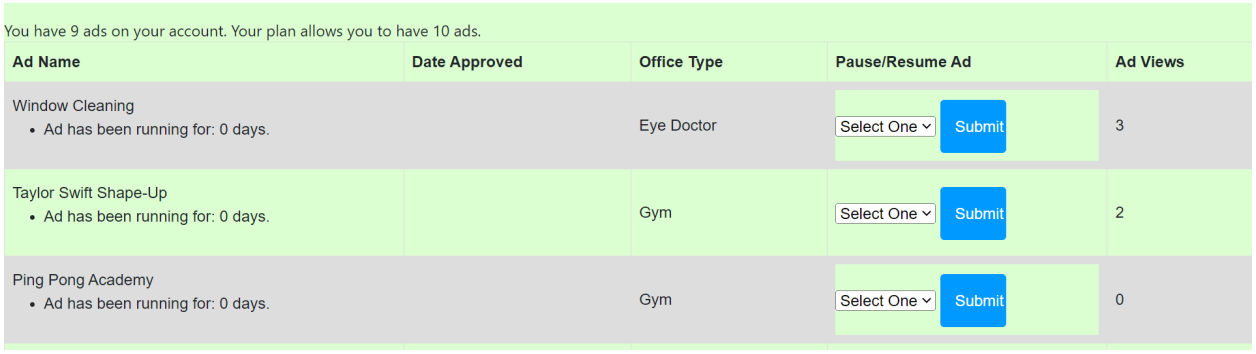
-Using the “value” of each option, the PHP code in my page will decide which business type category belongs to the ad due to the same naming convention in our BusinessType table.

```
<label for="business_type">Choose a business type to advertise to:</label>
<select id="business_type" name="business_type" default='Select One'>
    <option selected disabled>Select One</option>
    <option value="1">Dermatologist</option>
    <option value="2">Mechanic Shop</option>
    <option value="3">Eye Doctor</option>
    <option value="4">Gym</option>
    <option value="5">Pediatrician</option>
    <option value="6">Bakery</option>
    <option value="7">Restaurant</option>
    <option value="8">Book Shop</option>
    <option value="9">Salon</option>
    <option value="10">Department Store</option>
</select>
```

```
MariaDB [p2]> SELECT * FROM BusinessType;
+--------+------------------+
| typeID | typeOfBusiness   |
+--------+------------------+
|      1 | Dermatologist    |
|      2 | Mechanic Shop    |
|      3 | Eye Doctor       |
|      4 | Gym              |
|      5 | Pediatrician     |
|      6 | Bakery           |
|      7 | Restaurant       |
|      8 | Book Shop        |
|      9 | Salon            |
|     10 | Department Store |
+--------+------------------+
10 rows in set (0.000 sec)
```

Requirement 13: After an ad is submitted, the user will be able to use the web UI to check for statistics and analytics on ad viewership.
Competition Rate: 100%
Implementer/Tester: Joaquin/Raad(Program Ad View Count)

-The Ad Management page allows a user to see statistics such as how many views the ad has and how long it has been running



The Ad Views part of the page, retrieves a count of each time an ad is displayed in the PLATFORM device.
In the ad display function, each time a ad is displayed, there is if statement that gathers it's index and increments a counter

```
# Function that changes the images
def cycle():
    global count
    if count == adcount - 1:
        canvas.create_image(0, 0, image=advertisements[0], anchor='nw')
        count = 0
        counters[0] += 1
    else:
        canvas.create_image(0, 0, image=advertisements[count + 1], anchor='nw')
        count += 1
        if count == 1:
            counters[1] += 1
        if count == 2:
            counters[2] += 1
        if count == 3:
            counters[3] += 1
        if count == 4:
            counters[4] += 1
        if count == 5:
            counters[5] += 1
    # Loop portion of the function
    # The number is the time, ie 10,000 is 10 seconds
    window.after(1000, cycle)
    schedule.run_pending()
```

However this initial Implementation was supposed to use a PIR sensor to get accurate view counts if a user is nearby via motion. But the PIR sensor malfunctioned , and gave many false positives.

This was the original code for ad view count before the sensor malfunction

```
# Function that changes the images with PIR
def cyclepir():
    global count
    if count == adcount - 1:
        canvas.create_image(0, 0, image=advertisements[0], anchor='nw')
        count = 0
        pir.wait_for_motion(timeout=5)
        if pir.motion_detected:
          counters[0] += 1
    else:
       canvas.create_image(0, 0, image=advertisements[count + 1], anchor='nw')
       pir.wait_for_motion(timeout=5)
       if ((count == 1) and (pir.motion_detected)):
        counters[1] += 1
       if ((count == 2) and (pir.motion_detected)):
        counters[2] += 1
       if ((count == 3) and (pir.motion_detected)):
        counters[3] += 1
       if ((count == 4) and (pir.motion_detected)):
        counters[4] += 1
       if ((count == 5) and (pir.motion_detected)):
        counters[6] += 1

    # Loop portion of the function
    # The number is the time, ie 10,000 is 10 seconds
    window.after(2000, cycle)
    schedule.run_pending()
```

```
def update_view_count():
    for i in range(len(filenames)):
        insertquery = "Update p2.Advertisements INNER JOIN p2.Images on
p2.Advertisements.imageID = p2.Images.imageID set p2.Advertisements.displayCounter =" +
str(counters[i]) + " where p2.Images.imageName =" + '\'' + filenames[i] + '\''
        iq = insertquery
        cursor.execute(iq)
        connection.commit()
```

This function is what takes the counter values and sends all the values for the current running ads to the database.
The update view count function is called into a schedule function which can be timed.

```
schedule.every(1).seconds.do(update_view_count)
```

Requirement 14: The web UI shall provide users with alternate viewing options including color palettes, font styles, and font choices.
Competition Rate: 100%
Implementer/Tester: Raad

The themeselect function gets the value from the drop down list and does a classlist.toggle on the css class that has matches that theme, and that changes the website.

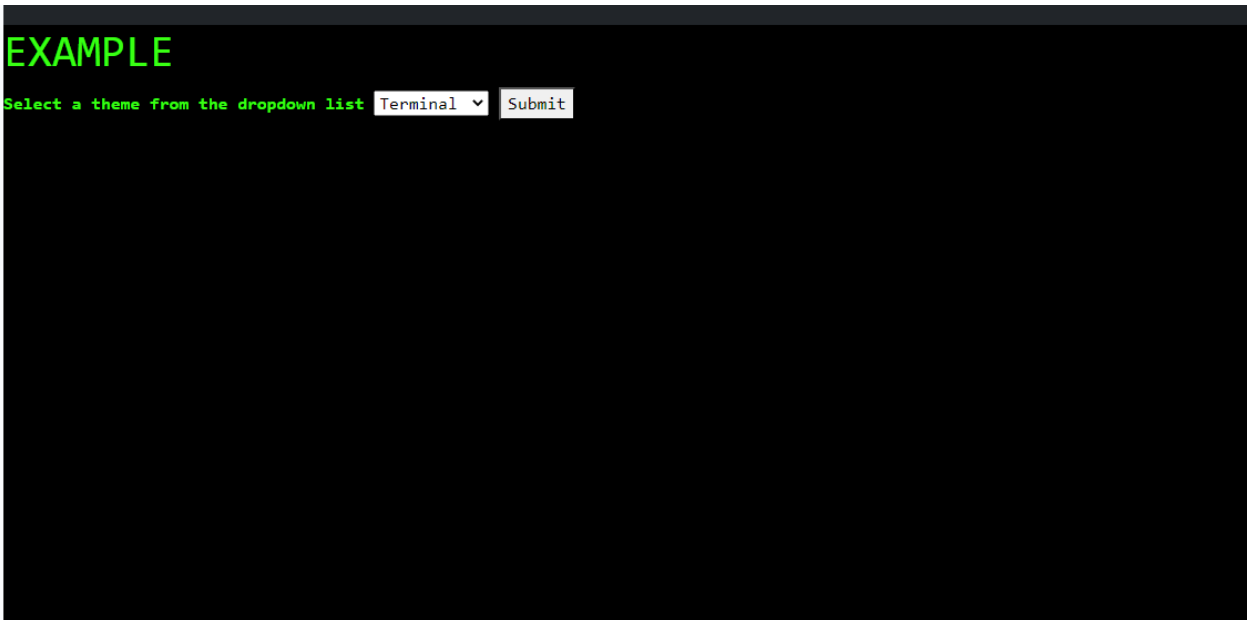
```
<!-- The local storage item stored from theme selection function is stored as string and able to toggle themes.-->
function currenttheme(){
curtheme= localStorage.getItem("theme");
var element = document.body;
string1 = localStorage.getItem("theme")
element.classList.toggle(string1);
}

<!-- Gets final click from selection menu and saves it as local storage item-->
function themeselect() {
var mylist = document.getElementById("themelist");
theme = mylist.options[mylist.selectedIndex].text;
localStorage.setItem("theme", theme);
}
```

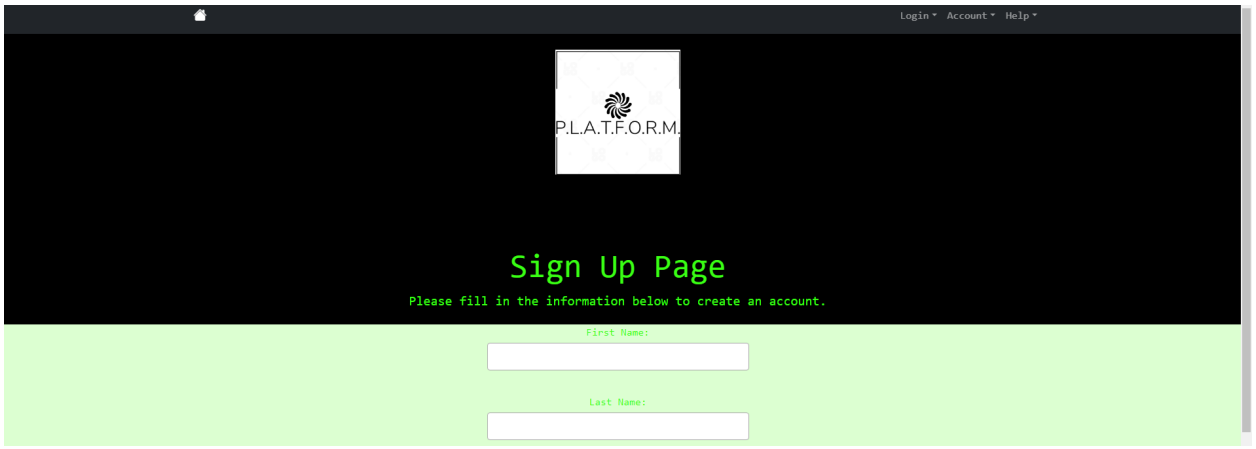
Each page has the current theme function and a reference to the css file,

```
    <script src="theme.js"></script>
</head>
<!-- On refresh the page goes to the saved theme option -->
<body onload="currenttheme()">
```

Onload the a saved local storage variable from the theme page is checked.


```
<!-- The local storage item stored from theme selection function is stored as string and able to toggle themes.-->
function currenttheme(){
curtheme= localStorage.getItem("theme");
var element = document.body;
string1 = localStorage.getItem("theme")
element.classList.toggle(string1);
}

<!-- Gets final click from selection menu and saves it as local storage item-->
function themeselect() {
var mylist = document.getElementById("themelist");
theme = mylist.options[mylist.selectedIndex].text;
localStorage.setItem("theme", theme);
}
```





Requirement 15: If a user has technical difficulties with the web UI there will be a self-help widget on the page to redirect the user to an issue report page.
Competition Rate: 70%
Implementer/Tester: Raad
On the support page, the html form section is done, however, I could not finish the PHP section. The normal PHP mail function requires a SMTP server to work, our team did not have that. Also if we want to have secure mail and be allowed into most email

services, the normal php mail function would not work, and we would have to install PHPmailer, which would require as to install a external library into our AWS ec2 VM.

Requirement 16: PLATFORM shall display ads as defined by the client.
Competition Rate: 100%
Implementer/Tester: Raad

This creates the fullscreen window, the ads are displayed in, it automatically gets the device resolution. The canvas section is the place in the window that displays the ads.

```
def create_window():
    global window
    global canvas
    global screen_width
    global screen_height
    # Set Screen Specifications, create root window
    window = tk.Tk()
    window.attributes('-fullscreen', True)
    screen_width = window.winfo_screenwidth()
    screen_height = window.winfo_screenheight()
    # Create a Canvas that has the images and is set in top of the root window
    canvas = tk.Canvas(window, width=screen_width, height=screen_height, highlightthickness=0)
    canvas.pack(fill="both", expand=True)
```

This is the function that displays the ads and loops through them through the whole day. The loop is cyclical, it starts at -1 increments up to the length of the advertisements array and restarts back to 0 to increment again.
The canvas function gathers the advertisements that were downloaded from the databases, converted into images, and put in the advertisements array, and displays it. The window.after function is the timing section of this code, every 1000 corresponds to 1 second. I used the tinkinter library for this display code. The tk.mainloop function is the function that starts the main function.

```
# Function that changes the images
def cycle():
    global count
    if count == adcount - 1:
        canvas.create_image(0, 0, image=advertisements[0], anchor='nw')
        count = 0
        counters[0] += 1
    else:
        canvas.create_image(0, 0, image=advertisements[count + 1], anchor='nw')
        count += 1
        if count == 1:
            counters[1] += 1
        if count == 2:
            counters[2] += 1
        if count == 3:
            counters[3] += 1
        if count == 4:
            counters[4] += 1
        if count == 5:
            counters[5] += 1
    # Loop portion of the function
    # The number is the time, ie 10,000 is 10 seconds
    window.after(1000, cycle)
    schedule.run_pending()
```

```
create_window()
# # # host1, port1, database1, user1, password1, namecolumnindex, datacolumnindex, plnamecolumnindex, pldatacolumnindex,  query1, query2, query3, filedir
retrieve_image("54.91.84.242", "3306", "p2", "remote", "db2023#",1,2, 1, 0, Queries[0], Queries[1],Queries[2], filedir)
create_counters()
schedule.every(1).seconds.do(update_view_count)
cycle()
tk.mainloop()
```

Changes From Original Proposal

Though there have not been many changes to our project since we made our original proposal for the P.L.A.T.F.O.R.M. system we do feel the need to list the following changes:

Requirement 5 required some adjusting to allow it to work in our current AWS environment. It was determined during development that the original requirement could not be achieved within the timeframe given, as we would require the use of an SMTP mail server within our AWS server to be able to send an email.

- The web UI shall provide users with a “forgot password” option to reset their password via email.

Was changed to be the following:

- *The web UI shall provide users with a “forgot password” option to unlock their account given accurate security question answers.*

Requirement 6 required some re-tooling to make it work within the constraints of our system. It was determined during development that the original requirement could not be achieved within the timeframe given.

- Users shall be able to pick a payment plan via the web UI, determining duration ads will show on PLATFORM devices.

Was changed to be the following:

- *Users shall be able to pick a payment plan via the web UI, determining the number of ads that they can upload to run on PLATFORM devices.*

Requirement 11 has been adjusted due to the constraints of our present environment. Due to the current design of the PLATFORM application and the structure of our database, as well as the fact that we presently only have one PLATFORM device for testing, we were unable to design the system to send different ads to different PLATFORM devices/locations. As such we have modified this requirement to accurately show how the system works at present.

- Moderators will approve ads submitted by users and send them to the appropriate PLATFORM locations as defined by the user.

Was changed to be the following:

- Moderators will approve ads submitted by users enabling PLATFORM devices to retrieve and display them as defined by the user.

Outside of the above changes there have been no other changes from the original proposal that are of note at this time.

User / Operation Manual

The PLATFORM system is a combination of both a physical display device that is located in doctor office examination rooms and other business' waiting rooms, and a web application that is used by advertisers to upload, manage, and publish their advertisements to the many locations where the devices are available. The physical devices are connected via WiFi internet and are in constant communication with the database and web application, allowing for clients to upload, and publish their ads through a simple user interface and also view metrics regarding their advertisement reach based on the locations and how often those ads are seen by customers in those different locations.

The following information covers basic user and moderator features of the system, explaining quickly and directly how to use the system, from account creation, to ad review, and finally publication of the advertisements.

Account Setup Page

- **Setting up an Account**
  The account sign-up page, which is called Sign Up in our navigation bar's Login dropdown folder, is primarily used to input various information to help in the creation of an account. To navigate to the account setup page, the user must use the navigation bar at the top of the home page and click the 'Sign Up' option under the 'Login' navigation section. To create an account, the user must fill out all of the input values according to the given parameters written in red below the input fields (ie. "Password must be at least 8 characters long"). When writing your inputs, as long as the requirements (the parameters written in red) turn green, your input is valid, and therefore it is a usable username/password. Once the user inputs a valid username and password, they must also choose two security questions from the dropdown box and type in answers to those questions. These answers, alongside the username and password, should be secretive, but also something you can easily memorize for later use. Both the password and the security question answers will be stored securely in our backend server. After completing all of this, the sign up button will change from being a translucent button to an opaque button, denoting that you have completed everything on this page. Lastly, in order to finalize your account set-up, click on the "Sign up" button. You will then be redirected to the Login page of the website.

Login Page

- **Login to Account**
  The login page is primarily used to input previously used information (ie. the username and password) to properly access the user's account. To navigate to the login page, the user must use the navigation bar at the top of the home page and click the 'Existing Users' option under the 'Login' navigation bar folder. To access your account using the

Existing Users page, the user must input their accurate username and password, keeping in mind that the username must be in email format. After doing so, the user must click the login button. If the user has input accurate credentials, then they are redirected to their unique Profile Management page. On the other hand, if the user has entered any inaccurate credentials, then an error will appear clearly stating what the issue is, for example: “Username does not exist. Please check your spelling.”. The user must correct the inaccuracies, and once that is done, they must click the Login button to be sent to their Profile Management webpage.

Forgot Password Page

- **Initiate the forget password process.**
  If a user has forgotten their password to access their account, the “Forgot Password?” link, at the bottom of the Existing Users page, will allow for users to change their password. To initiate the forget password process, the user inputs their email and clicks the submit button. Afterward, the user will be prompted to answer both of their security questions. If the answers entered by the user match the answers that are associated with this particular account, then the user will be redirected to a page that allows them to update the password on their account with a new one. Once the user submits this new password, they will be redirected to the Existing User page where they can attempt to login to their account again.

Account Lockout Page

- **Getting user’s account locked out**
  The account lockout page is a temporary page where the user is redirected to if they input the wrong password 4 times in the Existing Users login page. As noted earlier, the user is navigated to the account lockout page only if they fail to input the accurate username/password combination 4 times. At this page, they are prompted to answer both of their security questions. They must answer their security questions accurately, and if they do so, their account is unlocked and they are given their password. If they answer the questions incorrectly, their account stays locked and they must contact customer support for further assistance. To contact customer support, refer to our Support Form which is located in the FAQ webpage under the navigation bar’s Help folder.

Profile/Billing Management Page

- **Updating your information**
  On this page the user is prompted to enter their billing information so that they can provide the necessary information to sign up for a billing model on the next page that allows for them to upload ads for publication on the PLATFORM devices. The user is asked to enter their first and last names, their address information, the name on their credit card as well as the credit card number and security code/CVV number on the back of the card. Once this information is completed, they can click the submit button and move on to the Advertising payment plan page.

Advertising Pricing/Payment Plan page

- **Selecting a Payment Plan**
  This page shows new and returning users the three available pricing plans for the PLATFORM service. There is a Basic plan allowing for 2 Ads to be uploaded by a single user, a Standard plan allowing for 5 ads to be uploaded, and a Premium plan that allows for 10 ads to be uploaded by a single user. The user must select one of the options to move forward at this point as signing up is required to be able to upload any ads at all. Once a billing plan is chosen and the Sign Up button is clicked for that plan, the user will be redirected to the Advertisement Campaign Page.

Advertisement Campaign Page

- **Generate a new advertisement campaign**
  In the Ad Campaign Management page, which is also located in the navigation bar's Account drop down menu, the user must click the "Add New Ad Campaign" button to be redirected to the "Add New Ad Campaign" form. On the form the user must input the name of their new advertisement campaign, upload a PNG of the advertisement, choose the business type that this ad should display inside of, and then click on the reCAPTCHA. Clicking the Submit button will redirect the user to their Ad Management page, where they can see/update information regarding all of their submitted ads.

- **View ongoing ad campaigns**
  The advertisement campaign page, which is reached by going to the Ad Management web page listed under the navigation bar's Account drop down menu, allows users to view ongoing campaigns, pause or resume specific ads, and it shows analytical information for each individual advertisement..

Theme Page

- **Selecting a theme**
  On this page the user can select from a variety of themes that change color and font options for the website. To access this page, a user must click on the UI Themes link that is located in the navigation bar's Account drop down menu. To change themes, the user must make a selection through the dropdown box. Clicking the Submit button will cause the website's background/font color to change. Now the user can navigate the site with the selected theme maintained throughout their user experience.

Moderator Review Page

- **Advertisement review and approval by Moderator**
  When the user has completed the process of uploading an advertisement to the website, the ad is then put in the queue for moderator review before it can be displayed on any PLATFORM device. Once an ad has been uploaded to the database and flagged as needing moderator review, it automatically shows up on the moderator dashboard as a thumbnail image. A moderator can select the ad for review by clicking the thumbnail,

which opens a new page and displays a larger version of the ad as well as several different checkboxes that the moderator clicks, indicating whether the ad passes different aspects of the review process. If the ad passes review, the moderator checks the “passes review” radio button, indicating it has passed and clicks the “submit” button. This then marks the ad as being ready for display on the user selected PLATFORM devices. The database is updated accordingly and the ad is run on the following business day on the PLATFORM devices that fit the business type that user selected during the new ad creation process.

If the ad does not pass the review process, the moderator clicks the “does not pass review” radio button. At this point the moderator marks one of several available checkboxes as to why the ad did not pass review and can fill out specifics in a text box about why the ad did not pass review and what corrections would be needed to run the ad. Once this information has been entered the moderator clicks the “submit” button, and the ad is updated accordingly and the ad is removed from the database. The website automatically emails the user to notify them and provide the feedback regarding why the ad did not pass review. Once all corrections are made to the ad and it is resubmitted, it starts the Moderator Ad Review process over again.

Support Page

- **Using the frequently asked questions(FAQ) section**
  The FAQ section, which can be found in the FAQ webpage that is located under the navigation bar’s Help folder, details general information of the PLATFORM and advertisement service. Here users can access step by step guides to do various processes around the website. Users should utilize the support form for specified issues or concerns with feedback.

- **Using the support form**
  The support form, which can be found at the bottom of the FAQ webpage under the navigation bar’s Help folder, allows users to directly contact website admins for specific issues or questions and receive a response that will mediate it. The support form includes 3 text boxes where the user must input their first and last name along with a description of their problem. The form is submitted and the user can receive notification via email on the status of their request.

Sunny Day Video

Drive Link
https://drive.google.com/file/d/10JGU-5c_cxMLtOD0kzPkvre5BFnFectK/view?usp=share_link

Youtube link in case that does not work
https://youtu.be/inAjSyWKEKY

Self Evaluation by Stage

Stage	Level of Satisfaction
2	Met Expectations
3	Met Expectations
4	Met Expectations
5	Met Expectations
6	Exceeds Expectations
7	Exceeds Expectations

Self Rating of Project

What is the best part of the project that the team is proud to show off?

The website for our project, which is located at http://54.91.84.242/index.html, required all of us to come together and brush up on our HTML/CSS/JavaScript skills, but it also allowed us all to get hands-on experience working with PHP.

What is the most successful part of the project that allowed the team members to learn the most?

The team learned the most by utilizing a LAMP stack to create our project. Being able to integrate our MySQL database with our website, using the PHP skills we learned, was an arduous process, but we all feel like we gained some useful knowledge.

Which part of the project did the team dislike the most?

The last week of this project was the most stressful for many of our team members. With finals approaching, and the deadline for our project being so close, we all really pushed ourselves to make sure we were delivering on the product we set out to make 16 weeks ago.

Individual Assessments

Team Member: Joaquin Merida

Average Hours per Week: ≈ 5-8 **Total Hours**: ≈ 96

Contributions:

1. Came up with the original idea and the name of the project.
2. Set up our Amazon Web Service (AWS) account
    a. Which utilized S3, EC2, IAM Roles, and Amazon Linux 2022.
3. Set up the databases / tables in MySQL
4. HTML / CSS / PHP coding on the site
5. Provided some of the hardware necessary for the P.L.A.T.F.O.R.M.

Objectives	1	2	3	4	5	Total
Contribution 1	33%			33%	33%	100%
Contribution 2			33%	33%	33%	100%
Contribution 3		33%		33%	33%	100%
Contribution 4	33%	33%	33%			100%
Contribution 5	33%	33%	33%			100%
Total	100%	100%	100%	100%	100%	

5 Courses:

1. Advanced Database for IT - COP 4703
2. Hands-on Cyber Security - CIS 4622
3. Network Security and Firewalls - CNT 4403
4. Web Systems for IT - CGS 3853
5. Data & Database Security - COP 4931

Version 2 of P.L.A.T.F.O.R.M.
In version 2, I would like to improve the look of the physical components of the P.L.A.T.F.O.R.M.

Right now, the Raspberry Pi is secured to the monitor using velcro and wires are coming out of the hardware components in every direction. Ideally, version 2 of the P.L.A.T.F.O.R.M. would have the Pi housed inside the casing of the monitor in order to limit access to the device (for security reasons) and to look more visually appealing.

3 Suggestions for the Instructor:

1. I think it might be a cool idea to use the presentation time, at the end of the semester, as a type of networking event. If we're presenting in the Hall of Flags, would it be possible to get some industry professionals in there, during our presentations, in order to make some new connections or show off our projects?
2. I understand the need for documentation throughout the different phases of a project, but this was my first time getting introduced to documentation, on this kind of level, and it seemed like a lot of it was just busy-work. I would have preferred to spend a little less time on documentation and more time on working on my project.
3. More restrictions on the types of projects that a team can choose. I knew from the get-go that I didn't want to half-ass my senior project. So I found a team that was on the same page and wanted to do something impressive that utilized a lot of the skills we've learned over the last couple years. I noticed that some students in our class decided to do simple projects like a "password checker." It just seems like these students are doing themselves a disservice and not really applying themselves. At the end of the day, I don't really care what they do because it's obviously their choice, but I just don't think that simple projects really contribute to a student's overall growth, especially right before they're expected to graduate.

Personal Advice to Future Students:

It doesn't matter what project you decide to do, if you don't have the right team for your project, regardless of the project's difficulty level, it will not be a pleasant experience. So find members that you feel like you can trust!

I was lucky enough to find people who wanted to apply their prior knowledge, and put in the effort into learning new skills, in order to pull off our project. I've got no complaints.

Team Member: Abhiram Nelluri

Average Hours per Week: ≈ 5-8 **Total Hours**: ≈ 95

Contributions:

1. Helped streamline the project requirements: Accomplished objective 4, as when Joaquin brought this project idea up, I was able to assist in streamlining how it should be done and what ideas were in the realm of possibility for college students with a limited amount of technology and money could possibly implement.
2. Created all login functionality for the website: Accomplished objective 1 and 2, as I used ideas covered in the IT curricula (like HTML, JavaScript, and SQL) and not covered in the curricula (PHP). I was able to take my limited learned knowledge of those subjects to draw reasonable conclusions on how to connect HTML, JavaScript, and PHP to communicate with a database. It took some trial and error, but in the end I was able to formulate an analytical plan to properly create a login page that would achieve all the requirements I set, using the limited knowledge, internet, and inferences I could make.
    a. login.html
    b. login.js
    c. login-db.php
3. Created all account setup functionality for the website: Accomplished objective 1 and 2 (similarly to the login page), as I used ideas covered in the IT curricula (like HTML, JavaScript, and SQL) and not covered in the curricula (PHP). I was able to take my limited learned knowledge of those subjects to draw reasonable conclusions on how to connect HTML, JavaScript, and PHP to communicate with a database. It took some trial and error, but in the end I was able to formulate an analytical plan to properly create a setup page that would achieve all the requirements I set, using the limited knowledge, internet, and inferences I could make.
    a. setup.html
    b. setup-db.php
    c. signup.js
4. Created all account lockout functionality for the website: Accomplishes objective 3 and 5, as I was able to create various HTML and PHP pages based on experience with the login and setup pages, while adding additional functionality (that I learned through analyzing my previous code) to make the account lockout pages more efficient and simpler. On top of that, I was able to effectively function as a member of the team by regularly coming to meetings and sharing the updates to the functionality, so that the rest of the team were aware of what I was

working on, and we could figure out how we could improve it and make it mesh with the rest of the code that the team was working on.
    a. Check-account.php
    b. lock-account.php
    c. lockout.html
5. Created all account recovery functionality for the website: Accomplishes objective 3 and 5 (similarly to the account lockout pages), as I was able to create various HTML and PHP pages based on experience with the login and setup pages, while adding additional functionality (that I learned through analyzing my previous code) to make the account recovery pages more efficient and simpler. On top of that, I was able to effectively function as a member of the team by regularly coming to meetings and sharing the updates to the functionality, so that the rest of the team were aware of what I was working on, and we could figure out how we could improve it and make it mesh with the rest of the code that the team was working on.
    a. recovery.html
    b. security-questions.html
    c. security-questions-db.php

Objectives	**1**	**2**	**3**	**4**	**5**	**Total**
Contribution 1				100%		100%
Contribution 2	50%	50%				100%
Contribution 3	50%	50%				100%
Contribution 4			50%		50%	100%
Contribution 5			50%		50%	100%
Total	100%	100%	100%	100%	100%	

5 Courses:

1. COP 4703: Advanced Database for IT
2. CNT 4403: Network Security and Firewalls
3. CGS 3853: Web Systems for IT
4. CNT 4104: Computer Information Networks for IT
5. CEN 3722: Human Computer Interfaces for IT

Version 2 of P.L.A.T.F.O.R.M.
In version 2, I would like to improve the P.L.A.T.F.O.R.M. website's design and coding.

While the website works as intended, there were a lot of hiccups in its creation. One big thing that would be improved in version 2 would be the streamlining of code, so that it is easier to read and understand, and much more straightforward to implement. On top of that, I would also like to streamline all of the CSS styling, so that the website has a much more appealing and welcoming aesthetic.

3 Suggestions for the Instructor:

1. If possible, I would've liked to have a greater understanding of the timeline for our assignments ahead of time, so that planning would be much easier. While the syllabus gives a very general overline of when what will happen, I think if you were to give all the students a little bit more comprehensive a schedule/timeline, they would be able to plan their approach a lot better.
2. I would have preferred if there was a level of limitation or restriction on what projects were allowed/not allowed. This is primarily because my team and I put in a lot of time and effort into doing our project and are incredibly happy with the outcome. However, it feels like some teams chose very basic and easy to implement topics that might even be able to be found through a google search. I would appreciate some sort of requirement so that the future teams have to do something proper for their final college project.
3. I would have preferred to do far less documentation. While I understand the need and importance of documentation, I believe that having as many documents as we did was a little bit too much, especially for people who were still taking a full course load this semester. I propose that rather than have multiple documents due throughout the year, you have a single document that will be due every week or 2, that the students must constantly add on to. They would also have to have some sort of table early in the document that shows documentation information (version number, primary editor, etc.).

Personal Advice to Future Students:
There's a lot of advice I could give future students, but I think I will focus on the 2 things I consider most important: time management and teammates.

It is of vital importance that you understand how much time you have before each due date, and allocate work accordingly, as if you don't you could possibly end up not doing more than half of your project. Be very cognizant of the time you are putting into each task, so that you can complete everything within the time available.

More importantly, however, is that you choose a team that you can rely on and put your trust in. Admittedly, I didn’t know any of my teammates prior to this class, so I was a little wary of them being lazy and not working, but thank god I was proven wrong. Your teammates are what make this class much more easy and bearable, so choose people you are willing to work with and stick through it with them.

Team Member: John Liegl

Average Hours per Week: ≈ 5-8 **Total Hours**: ≈ 96.

Contributions:

1. Helped crystalize and structure how best to implement the project.
    a. Joaquin came up with the original idea for PLATFORM, and when proposed we then discussed how it could be implemented as a group. Specifically I proposed the usage of a PIR sensor which got added into the final design spec as a way to control when the device was on and off and also a way to track actual viewership numbers for analytics.
2. Took meeting notes, led meetings, and maintained the LiveBinder website.
    a. As stated above, while I served as the original group team leader I continued my meeting note taking and LiveBinder maintenance processes for the entire duration of the class and project.
3. Designed portions of the website using HTML, CSS, JS, PHP and SQL.
    a. All members participated to one degree or another in this process. In particular I designed the website address/billing information page, the reCAPTCHA coding for image submission, the payment plan page, and both pages of the Moderator ad/image approval functionality. 90% of the coding I implemented interacted with PHP and the MariaDB MySQL database in one way or another.
4. Researched application design platforms that could be used to build our application that ran on Raspberry PI and could access the MariaDB we implemented.
    a. During the project progression it became apparent to me that we were so focused on getting the website and database working that we had not considered which programming language or app development tool we were going to use to develop the app to display the ads on our Raspberry PI. I looked into a number of apps that would work with our technology stack and provided that information to the group.
5. Researched, proposed ideas, and discussed how best to implement our website and the database necessary for the project to function according to specification.
    a. Specifically I initiated the process of finding a way for us to work collaboratively by looking into utilizing GitHub for our coding and website hosting. I also researched using Azure as a possible alternative to host our website/server for the database as it would have worked with our other chosen languages and hardware. However, during discussions we were presented with AWS as a superior alternative which I agreed to given our discussions at the time and what the group wanted. I therefore dove in

headfirst to learn and master AWS' S3 buckets and VM hosting along with our team as a whole.

Objectives	1	2	3	4	5	Total
Contribution 1	33.3%			33.3%	33.3%	100%
Contribution 2				33.3%	66.6%	100%
Contribution 3			66.6%	33.3%		100%
Contribution 4	33.3%	33.3%	33.3%			100%
Contribution 5	33.3%	66.6%				100%
Total	100%	100%	100%	100%	100%	

5 Courses:

1. COP 4703 - Advanced Database Systems for IT.
2. CGS 3853 - Web Systems for IT.
3. COP 3515 - Advanced Program Design for IT.
4. CNT 4104 - Computer Info Networks for IT.
5. CEN 3722 - Human Computer Interfaced for IT.

Version 2 of P.L.A.T.F.O.R.M.

If we were to design Version 2 of P.L.A.T.F.O.R.M I would like to re-design the organization and structure of our website coding. While we were able to get things very organized and streamlined around the time we implemented the database and started using PHP, we were much more independently working and using our own design aesthetics during the initial website creation. I would have us spend a lot more time designing, thinking about, and coming together around which frameworks (Bootstrap or otherwise) we wanted to use for the HTML code, and also come to a consensus about what we wanted the website to look like. As it is, the website looks ok, but with more time and collaboration we could have made it look a lot more modern.

I would also like to improve the functionality of the PIR sensor and get the bugs worked out of that part of our physical device. It works well, intermittently, and so I would like to spend time working on getting it to be more reliable and to provide additional functionality, such as controlling the on/off state of the device as a whole. Right now it

serves as a decent tracker to note when there are people present seeing the ads, but that is all it does, and it only does that when it is working correctly.

Finally, if I were to enhance our services, I would want to implement an SMTP server so that we wouldn't need to rely entirely on security questions to unlock an account that is locked due to too many failed login attempts. Security questions are notably weak as a security measure, so having a user need to answer them, then receive an email to their registered email account to unlock everything would be a good way to better secure client accounts. Also, it would allow for users to ask direct questions to the staff/moderators of the service and get directed answers back, and it would enable the moderators to be able to provide specific feedback about any advertisement that did not pass the review process, giving the clients an opportunity to correct the ad and have it re-reviewed.

I would also want to enhance our services by implementing a 3rd party billing service into our website such as PayPal, Clover, Stripe, etc. Doing this was outside the scope of our project as most of these services require that companies pay to utilize their API and services. If we were to continue and turn this into a business I would want to move away from any kind of payment card info and the security problems related to it.

3 Suggestions for the Instructor:

1. While having more time to work on the project was great, I think having at least one (optional) meeting in each project stage would be useful. This way students who are less familiar with concepts like UML, Requirements & Specification, Analysis, etc. would have the opportunity to be presented with basic information and ask questions to help them understand the requirements of the stage.
2. I think some kind of Wiki, or help page would be useful to collect all the sources and topics that students would need to know to be successful in their project. While this information was provided in various assignment instructions, having it all together, broken down by project stage/type of information would be very useful for new students who are unsure how to proceed.
3. I think that directing the students to begin developing their projects earlier would be very useful. The first 4 weeks are very report/analysis heavy and there is little motivation given to students to begin coding until Stage 5 which does not leave much time to develop a decent prototype. While the reports done in the first 4 stages are important, and can help shape the actual project, students should at least be encouraged to plan out the broad strokes, such as coding languages to be used, services that will be implemented, and coming up with a shared standard for how code is to be written, etc.

Personal Advice to Future Students:
Meet, and meet often. There will be disagreements, there will be misunderstandings and confusion at times from all members of the group. But if you are in constant communication about what you are all doing, what you are working on, what you think will or will not work, then everything will sort itself out in the end. If you are not meeting and discussing what you are working on and where you are heading, you are bound to have a lot more re-work to do at the end of the day.

Also, it is extremely important that when doing the initial development stages that you work with your group to determine what your development process will be as a whole and get down terminology and coding protocols. If you do not discuss this (as is easy to do) you will find that while what you each do will work together, it may not appear seamless as different people have different design preferences. Work together on coding to ensure that you are all on the same page.

Team Member: Raad Chowdhury

Average Hours per Week: ≈5-8 **Total Hours**: ≈96

Contributions:

1. I came up with the idea of uploading BLOBS to the database, so the device hosting the PLATFORM program can access the ad images from the database. Also creating the prototype implementation of BLOB to database upload, which Joaquin incorporated to his ad upload page.
2. I created the PLATFORM program in its most basic form pulling ads from the database and playing it in timed loops throughout the day. The program is universal to different machines if they run python and have appropriate modules installed. The program is configured to the database, and only the file directory needs to be changed.
3. I came up with the scheduling mechanism idea and having only 6 ads play in the PLATFORM as part of its implementation. I also created the PLATFORM program implementation of it which utilizes the sql where clause to search for the current day the program is running it. I also created the prototype list form and php code of the schedule, which was fully implemented by Joaquin.
4. I tested the PLATFORM program on the raspberry pi and pir sensor, which would be used as a test case on how the program would operate in a real world office environment.On the device, I worked on code that used the PIR sensor to wait for nearby motion, and increment a counter if motion is detected. This counter is then transported to the database and used as a statistic. However this PIR code was not implemented in the final version, because the sensor malfunctioned with false positives. So in its current form the counter just increments on each run of the ad.
5. I made the theme and FAQ pages on the website, however the support section of the FAQ page was not fully implemented due to the lack of a SMTP server.

Objectives	**1**	**2**	**3**	**4**	**5**	**Total**
Contribution 1		25%		50%	25%	100%
Contribution 2		25%		50%	25%	100%
Contribution 3			50%		50%	100%
Contribution 4	50%	25%				75%
Contribution 5	25%		50%			75%
Total	75%	75%	100%	100%	100%	

5 Courses:

1. COP 3515, Advanced Program Design for IT
2. COP 4404, Network Security and Firewalls
3. CGS 3853. Web Systems
4. COP 4703 Advanced Databases for IT
5. COP 4538 Data Structures and Algorithms for IT

Version 2 of P.L.A.T.F.O.R.M.

In version two, I want to focus on improving the PLATFORM program. I would have a more complex scheduling system, where you can target ads hourly, which I did not have the time to implement in our current iteration. I would also have different versions of the program that can be placed in different businesses and access different parts of the database. Since in our original proposal the program was supposed to have ads targeted to related businesses.

3 Suggestions for the Instructor:

1. Make the functional requirements flexible and less of an impact on the grade, since the scope of the project is changing, and the requirements were made early one before implementation.
2. Lessen the workload of the stage reports.
3. I think the final grade should take into account all requirements, not just functional ones.

Personal Advice to Future Students:

When writing the functional requirements,be realistic,do not think too far ahead. Before writing them make a plan on which goals you have the resources and knowledge to accomplish. Also start the hard things early, so it would be easier for you to accomplish your full vision.