Criterion C: Development

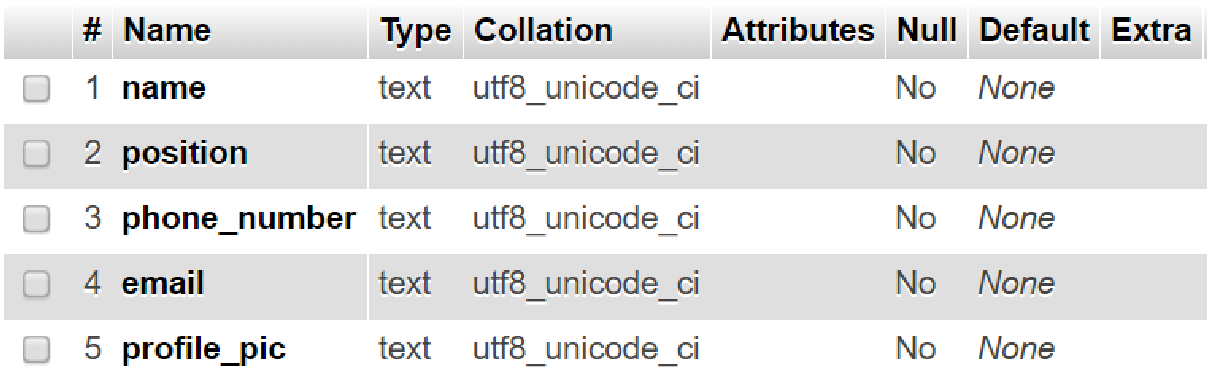
The following techniques are used in my internal assessment:

1. MySQL
    a. Table creation
    b. Data selection
    c. Data update
    d. Data insertion
    e. Data deletion
2. HTML
    a. Input type text and password
    b. Input type submit
    c. Input type file
3. PHP
    a. Automatic logout of the administrator panel and password protection
    b. Upload of a .csv file
    c. Download of a .csv file
    d. Notification sending
    e. Sending tables to the application

1. Techniques in MySQL

A. Creation of table

In order to use the tables, they firstly need to be created. This was executed in phpMyAdmin.

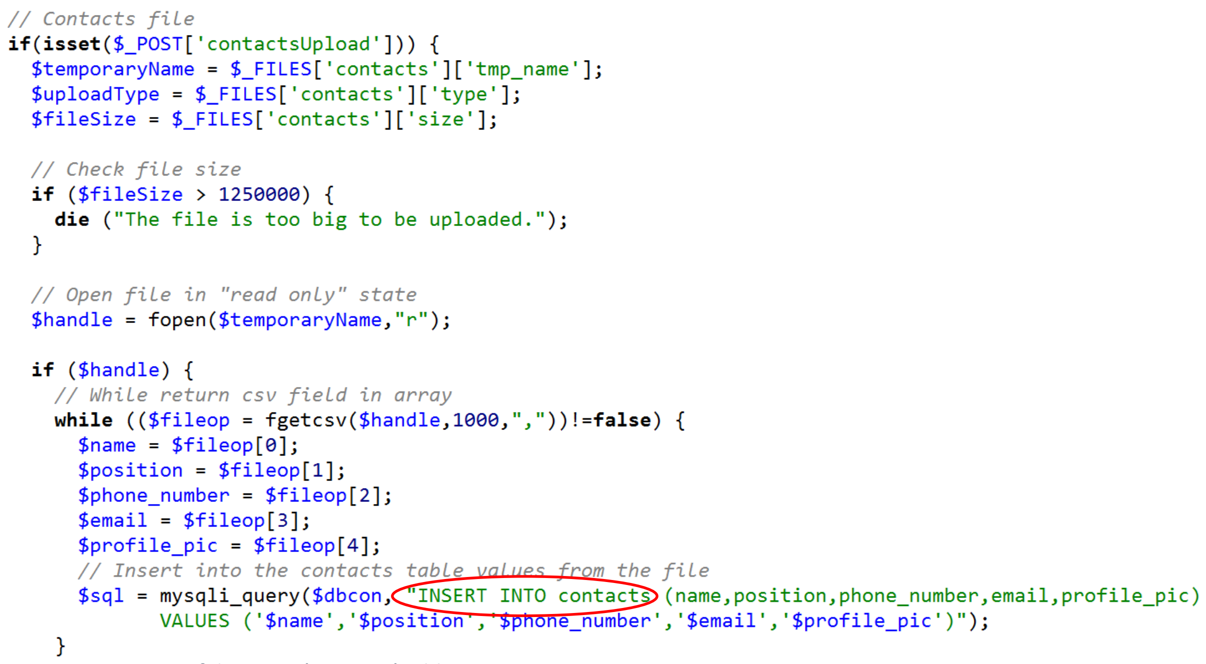


	#	Name	Type	Collation	Attributes	Null	Default	Extra
☐	1	name	text	utf8_unicode_ci		No	None	
☐	2	position	text	utf8_unicode_ci		No	None	
☐	3	phone_number	text	utf8_unicode_ci		No	None	
☐	4	email	text	utf8_unicode_ci		No	None	
☐	5	profile_pic	text	utf8_unicode_ci		No	None	

Figure 1. Structure of table 'Contacts' after it has been created in phpMyAdmin.

B. Data insertion

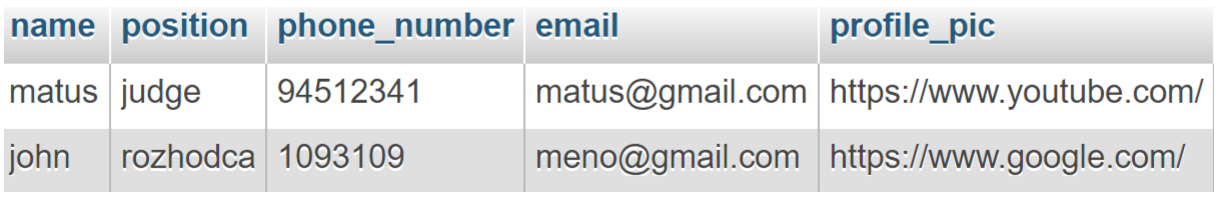
My client wanted to be able to upload a .csv file into the database. This means that new data needs to be inserted into the specific table. This is illustrated for table ‘contacts’ below.



```
// Contacts file
if(isset($_POST['contactsUpload'])) {
  $temporaryName = $_FILES['contacts']['tmp_name'];
  $uploadType = $_FILES['contacts']['type'];
  $fileSize = $_FILES['contacts']['size'];

  // Check file size
  if ($fileSize > 1250000) {
    die ("The file is too big to be uploaded.");
  }

  // Open file in "read only" state
  $handle = fopen($temporaryName,"r");

  if ($handle) {
    // While return csv field in array
    while (($fileop = fgetcsv($handle,1000,","))!=false) {
      $name = $fileop[0];
      $position = $fileop[1];
      $phone_number = $fileop[2];
      $email = $fileop[3];
      $profile_pic = $fileop[4];
      // Insert into the contacts table values from the file
      $sql = mysqli_query($dbcon,"INSERT INTO contacts (name,position,phone_number,email,profile_pic)
             VALUES ('$name','$position','$phone_number','$email','$profile_pic')");
    }
```

Figure 2. Insertion of data into 'contacts' table

Based on the content of the .csv file, the following result was achieved.

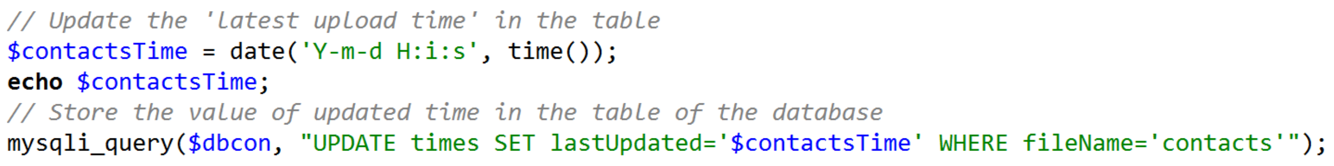


name	position	phone_number	email	profile_pic
matus	judge	94512341	matus@gmail.com	https://www.youtube.com/
john	rozhodca	1093109	meno@gmail.com	https://www.google.com/

Figure 3. Table 'contacts' after the data has been inserted.

C. Data update

Every time a file is uploaded into the database, the time of the latest upload is updated. The time values for all tables that can be uploaded are stored in one table called ‘times’. The below example demonstrates how time of ‘contacts’ table is updated after the new data into the ‘contacts’ table is inserted.

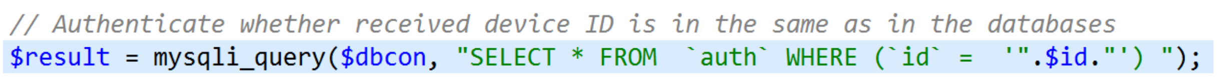


```
// Update the 'latest upload time' in the table
$contactsTime = date('Y-m-d H:i:s', time());
echo $contactsTime;
// Store the value of updated time in the table of the database
mysqli_query($dbcon, "UPDATE times SET lastUpdated='$contactsTime' WHERE fileName='contacts'");
```

Figure 4. Data update in table 'times'

D. Data selection

Any time data from the database needs to be sent to the application or displayed in the administrator panel, SQL command SELECT is used. In the example below, it is used to authenticate device ID and find out if the value of received ID is the same as in the database.

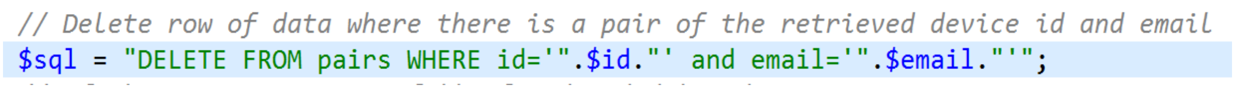


```
// Authenticate whether received device ID is in the same as in the databases
$result = mysqli_query($dbcon, "SELECT * FROM  `auth` WHERE (`id` =  '".$id."') ");
```

Figure 5. Selection of data from table 'auth'

E. Data deletion

Any time a contestant logs out of the application, his device ID as well as email contact should be deleted. This is used in logout.php.



```
// Delete row of data where there is a pair of the retrieved device id and email
$sql = "DELETE FROM pairs WHERE id='".$id."' and email='".$email."'";
```

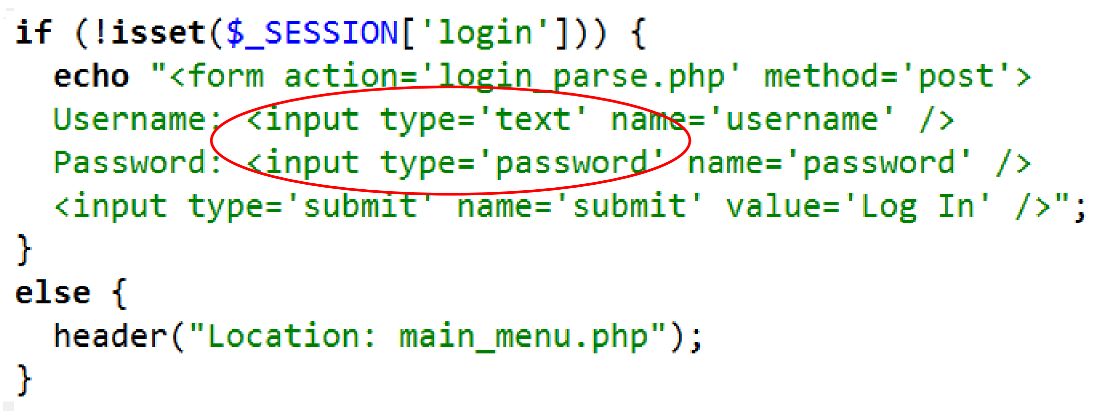
Figure 6. Data deletion in table 'pairs'

2. Techniques in HTML

HTML provides graphical user interface and creates platform to enable the user perform actions on the data transfer. Following input types of the platform were used:

A. Input type text and *password*

These input types are already used in the login page and create a text area where the user can type his username and password.



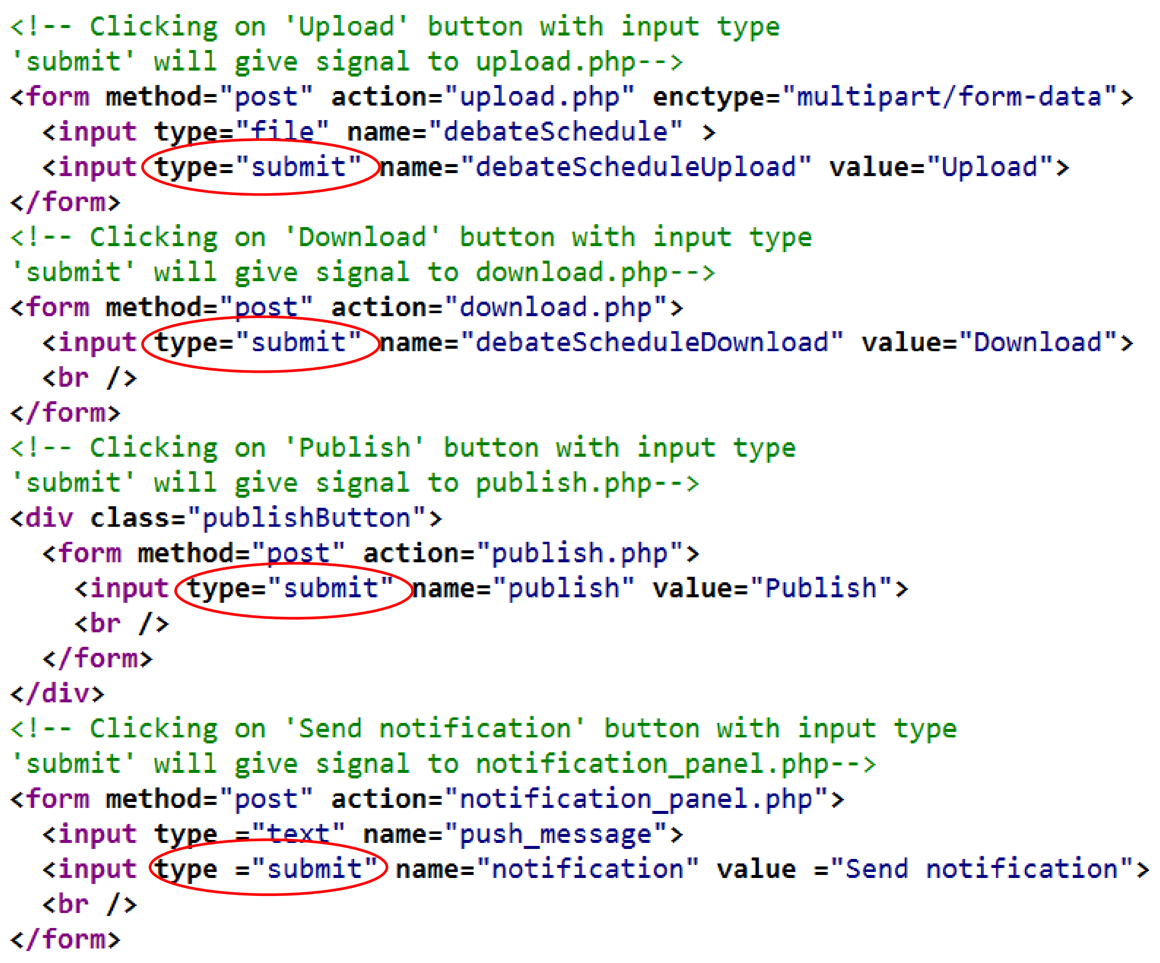
```
if (!isset($_SESSION['login'])) {
   echo "<form action='login_parse.php' method='post'>
   Username: <input type='text' name='username' />
   Password: <input type='password' name='password' />
   <input type='submit' name='submit' value='Log In' />";
}
else {
   header("Location: main_menu.php");
}
```

Figure 7. HTML code for the login page with input type 'text' and 'password'

To send a signal to the *login_parse.php* file a button of type *submit* needs to be clicked on. This is explained below.

B. Input type *submit*

When button with this input type is clicked on, it gives appropriate php file a signal to start operate. Figure below shows use of this input type in the main page where different buttons with this input type call different php files.

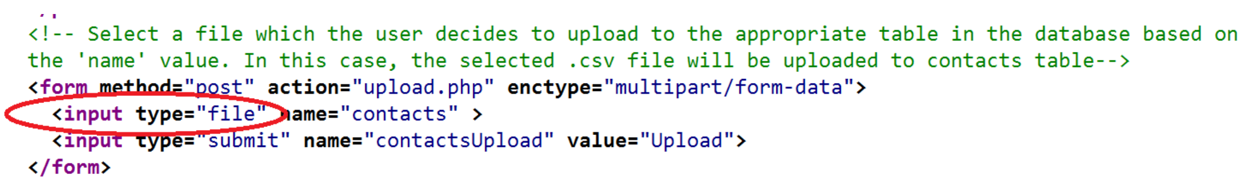


```
<!-- Clicking on 'Upload' button with input type
'submit' will give signal to upload.php-->
<form method="post" action="upload.php" enctype="multipart/form-data">
  <input type="file" name="debateSchedule" >
  <input type="submit" name="debateScheduleUpload" value="Upload">
</form>
<!-- Clicking on 'Download' button with input type
'submit' will give signal to download.php-->
<form method="post" action="download.php">
  <input type="submit" name="debateScheduleDownload" value="Download">
  <br />
</form>
<!-- Clicking on 'Publish' button with input type
'submit' will give signal to publish.php-->
<div class="publishButton">
  <form method="post" action="publish.php">
    <input type="submit" name="publish" value="Publish">
    <br />
  </form>
</div>
<!-- Clicking on 'Send notification' button with input type
'submit' will give signal to notification_panel.php-->
<form method="post" action="notification_panel.php">
  <input type ="text" name="push_message">
  <input type ="submit" name="notification" value ="Send notification">
  <br />
</form>
```

Figure 8. HTML code in the main menu with input type 'submit'

C. Input type *file*

This input type is used to select a file from the computer, in this case to upload a .csv file.



```
<!-- Select a file which the user decides to upload to the appropriate table in the database based on
the 'name' value. In this case, the selected .csv file will be uploaded to contacts table-->
<form method="post" action="upload.php" enctype="multipart/form-data">
  <input type="file" name="contacts" >
  <input type="submit" name="contactsUpload" value="Upload">
</form>
```

Figure 9. HTML code in the main menu with input type 'file

3. Techniques in PHP

PHP code serves as a communication channel between the front-end of the application and the database. It provides communication with the front-end part that David codes as well as with the administrator panel.

A. Automatic logout of the administrator panel after user's inactivity and protection of the webpage by user's login

```
// Start the session
session_start();
// Redirect the user into the login page if he has not logged in
if (!isset($_SESSION['username'])){
  header('location: index.php');
  exit;
}
```

Figure 10. Redirection to the login page (index.php) when the user has not logged in.

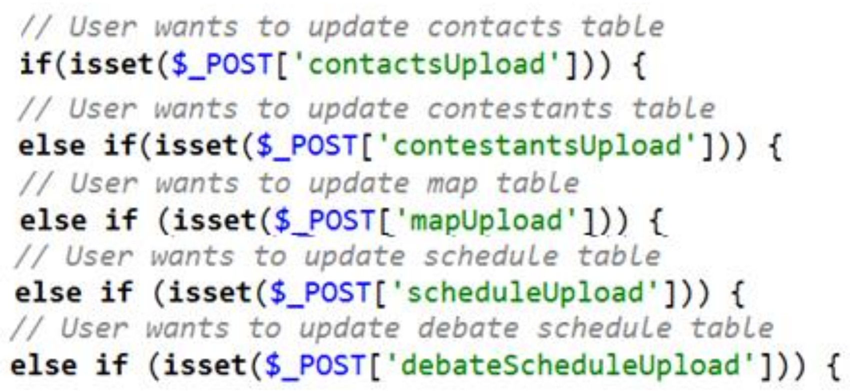
Time value is updated every time user performs an action. If he is inactive for more than 15 minutes, he will be automatically redirected to the login page next time he wants to do any action.

```
// If the user is inactive for the given time period, redirect him back to the login page
if(time() - $_SESSION['timestamp'] > 900) {
  header('location: index.php');
  exit;
}
// Update timestamp when the user performs an action
$_SESSION['timestamp'] = time();
```

Figure 11. Redirection to the login page (index.php) when the user has been inactive for 15 minutes

B. Upload a selected file to the database

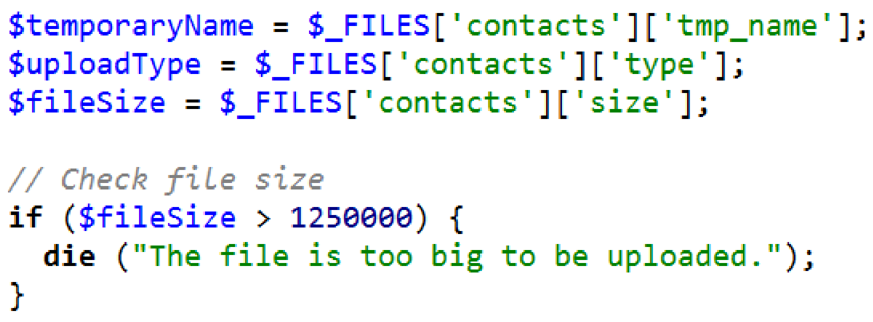
Upload of any .csv file selected in the administrator panel is provided by file *upload.php*.



```
// User wants to update contacts table
if(isset($_POST['contactsUpload'])) {
// User wants to update contestants table
else if(isset($_POST['contestantsUpload'])) {
// User wants to update map table
else if (isset($_POST['mapUpload'])) {
// User wants to update schedule table
else if (isset($_POST['scheduleUpload'])) {
// User wants to update debate schedule table
else if (isset($_POST['debateScheduleUpload'])) {
```

Figure 12. Determines where the signal comes from and which table should be uploaded

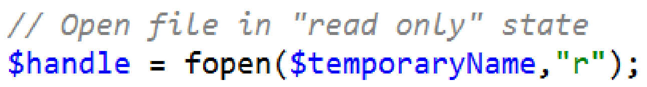
Uploading process is different for each table and depends on the number of the table's columns. Nevertheless, the scheme used is the same. Below is example of upload a .csv file into 'contacts' table.



```
$temporaryName = $_FILES['contacts']['tmp_name'];
$uploadType = $_FILES['contacts']['type'];
$fileSize = $_FILES['contacts']['size'];

// Check file size
if ($fileSize > 1250000) {
  die ("The file is too big to be uploaded.");
}
```

Figure 13. Store basic file information and make a control check of the file's size

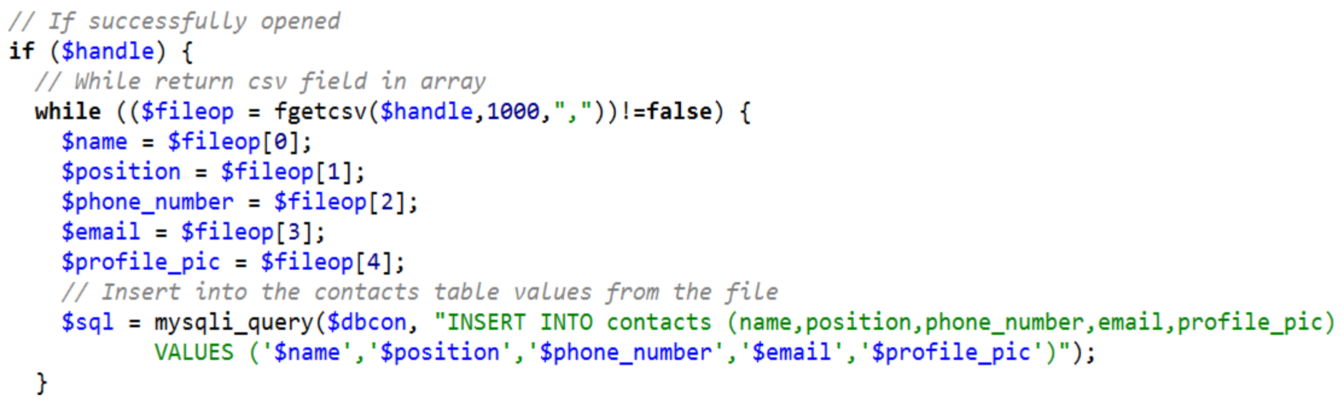
The process continues with opening the file in read-only mode.



```
// Open file in "read only" state
$handle = fopen($temporaryName,"r");
```

Figure 14. Open the file in 'read only' mode

If the file has been successfully opened, it will go row by row and store appropriate data into the characteristic variable based on the comma separation. This process stops when all data from the file has been searched through. The data is then inserted into the table.

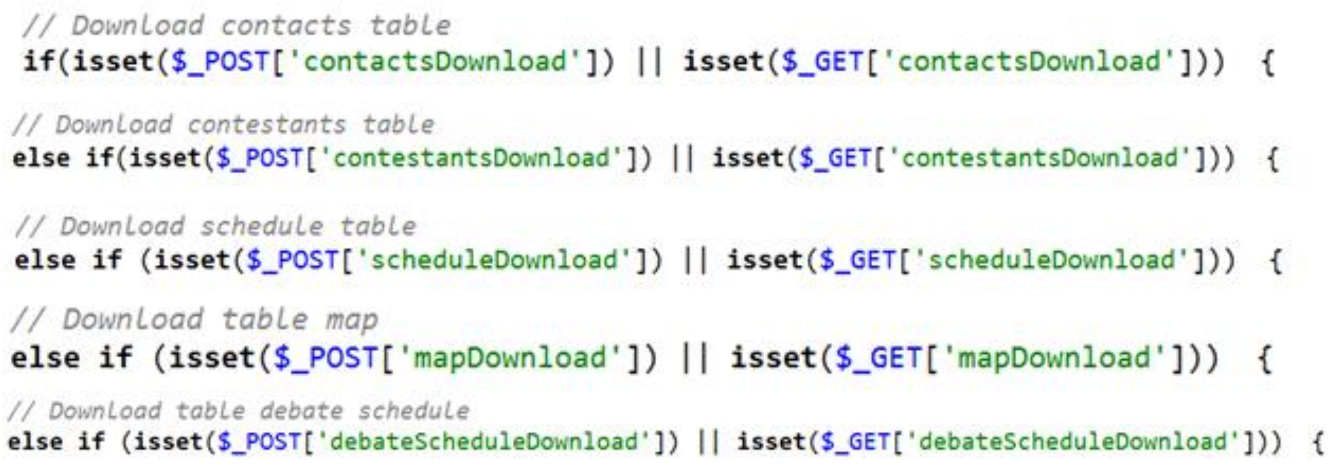


```
// If successfully opened
if ($handle) {
  // While return csv field in array
  while (($fileop = fgetcsv($handle,1000,","))!=false) {
    $name = $fileop[0];
    $position = $fileop[1];
    $phone_number = $fileop[2];
    $email = $fileop[3];
    $profile_pic = $fileop[4];
    // Insert into the contacts table values from the file
    $sql = mysqli_query($dbcon, "INSERT INTO contacts (name,position,phone_number,email,profile_pic)
           VALUES ('$name','$position','$phone_number','$email','$profile_pic')");
  }
```

Figure 15. Upload the .csv file into table 'contacts'

C. Download a .csv file from the database

The download of the .csv file into the computer was not successful. This might have been caused by an ambiguous file directory in my computer and/or firewall in the computer. After discussion with the user, he told me that it is enough for him when the file is downloaded but echoed as .csv file and thus can be easily copied into the computer. The process starts similarly as the upload:



```
// Download contacts table
if(isset($_POST['contactsDownload']) || isset($_GET['contactsDownload']))  {

// Download contestants table
else if(isset($_POST['contestantsDownload']) || isset($_GET['contestantsDownload']))  {

// Download schedule table
else if (isset($_POST['scheduleDownload']) || isset($_GET['scheduleDownload']))  {

// Download table map
else if (isset($_POST['mapDownload']) || isset($_GET['mapDownload']))  {

// Download table debate schedule
else if (isset($_POST['debateScheduleDownload']) || isset($_GET['debateScheduleDownload']))  {
```

Figure 16. Determines where the signal comes from and which table should be downloaded

Based on the user's selection, the table is processed as .csv file and outputted for the user.

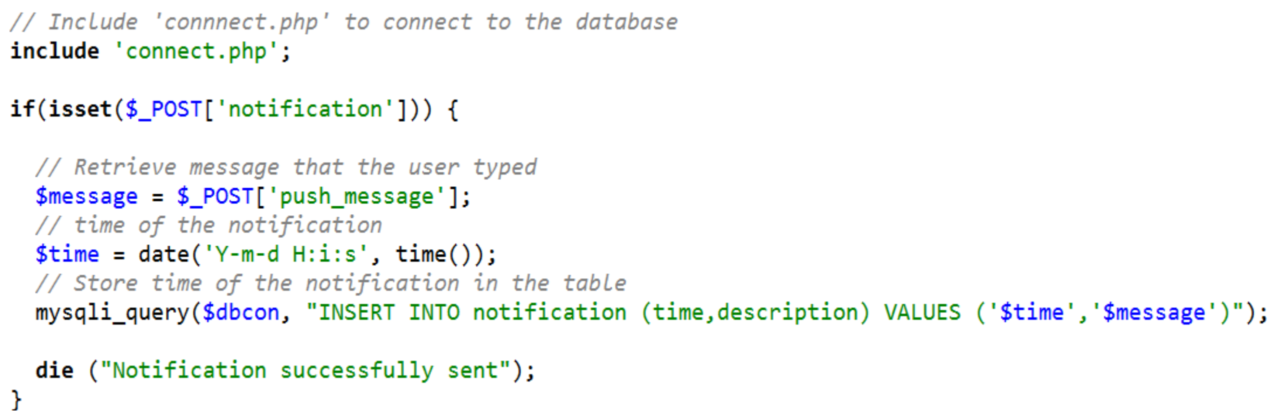
```php
// Select everything from the chosen table of the database
$result = mysqli_query($dbcon, "SELECT * FROM debateSchedule") or die('query failed');
// Get the number of rows of the table
$num_rows = mysqli_num_rows($result);
// Check if table is not empty
if($num_rows == 0) {
  die('Debate schedule table is empty');
}

// Retrieve the first row of the table as an associative array (headlines)
$row = mysqli_fetch_assoc($result);
// Open the file in "write-mode"
$file_open = fopen($filename,"w");
// Initialize the separators as empty
$seperator = "";
$comma = "";

// Go through the whole row
foreach($row as $name => $value) {
  $separator .= $comma . '' .str_replace('','""', $name);
  $comma = ",";
}

// Go to the next line
$separator .="\n";
// Write to the open file
fputs($file_open, $separator);
// Go to the next line
$separator .="\n";
// Write to the open file
fputs($file_open, $separator);

// Do this for the whole file
while($row = mysqli_fetch_assoc($result)) {
  $seperator = "";
  $comma = "";
  foreach($row as $name => $value) {
    $separator .= $comma . '' .str_replace('','""', $value);
    $comma = ",";
  }
  // Add a new line in the end
  $separator .="\n";
  fputs($file_open, $separator);
}

// The file has been successfully processed
echo $separator;
// Close the file
fclose($file_open);
```

Figure 17. Display table 'debateSchedule' in .csv format in the administrator panel

D. Send notification

The user can type a message in the administrator panel and after sending it, it will be inserted into the database. When there is a signal from the application, all notifications (messages) in the database which were inserted after the last time of this request will be sent into the application.

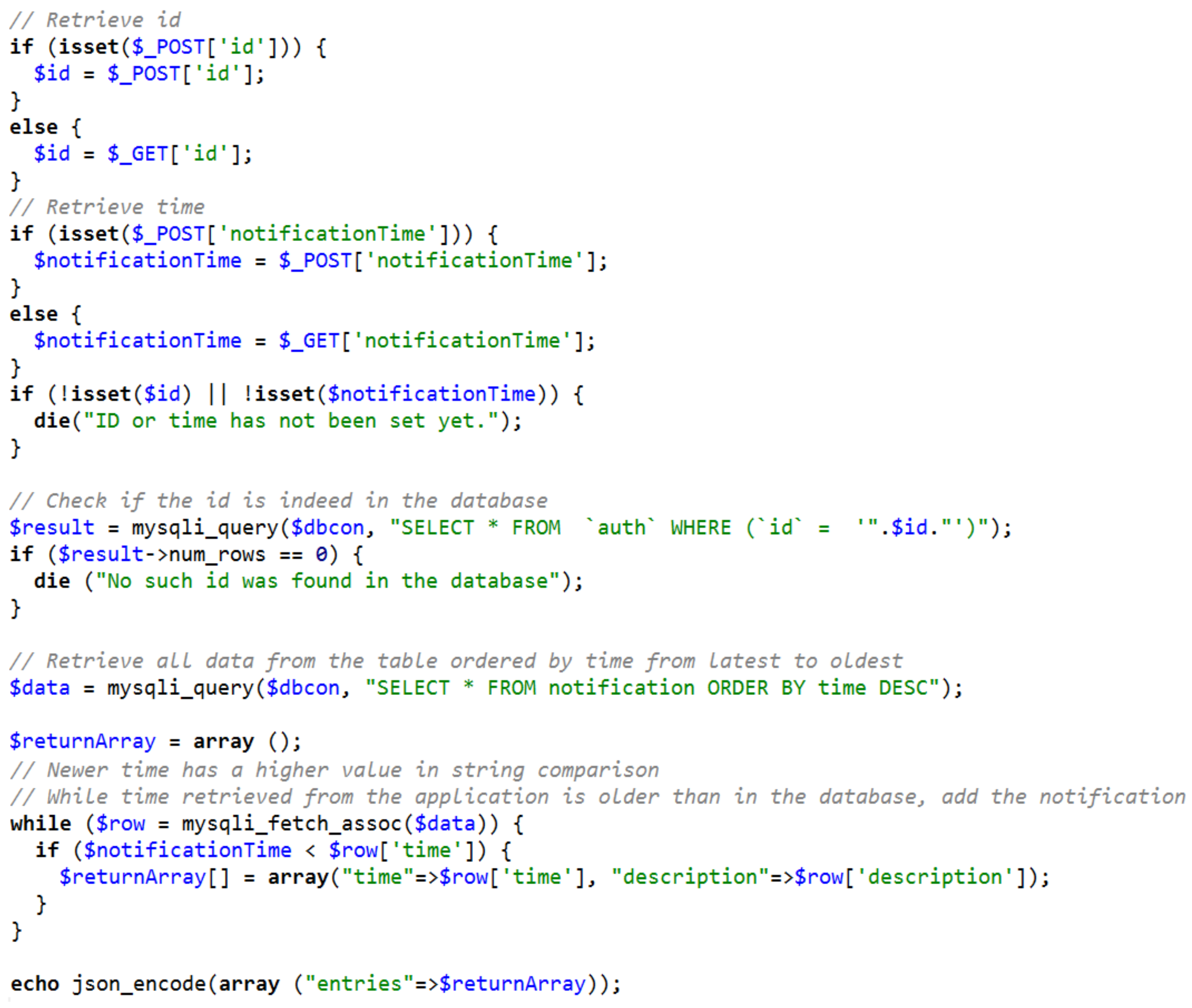
The code below gets the request from the administrator panel



```php
// Include 'connnect.php' to connect to the database
include 'connect.php';

if(isset($_POST['notification'])) {

  // Retrieve message that the user typed
  $message = $_POST['push_message'];
  // time of the notification
  $time = date('Y-m-d H:i:s', time());
  // Store time of the notification in the table
  mysqli_query($dbcon, "INSERT INTO notification (time,description) VALUES ('$time','$message')");

  die ("Notification successfully sent");
}
```

Figure 18. Store notification in the table

The code below handles communication with the application



```php
// Retrieve id
if (isset($_POST['id'])) {
  $id = $_POST['id'];
}
else {
  $id = $_GET['id'];
}
// Retrieve time
if (isset($_POST['notificationTime'])) {
  $notificationTime = $_POST['notificationTime'];
}
else {
  $notificationTime = $_GET['notificationTime'];
}
if (!isset($id) || !isset($notificationTime)) {
  die("ID or time has not been set yet.");
}

// Check if the id is indeed in the database
$result = mysqli_query($dbcon, "SELECT * FROM  `auth` WHERE (`id` =  '".$id."')");
if ($result->num_rows == 0) {
  die ("No such id was found in the database");
}

// Retrieve all data from the table ordered by time from latest to oldest
$data = mysqli_query($dbcon, "SELECT * FROM notification ORDER BY time DESC");

$returnArray = array ();
// Newer time has a higher value in string comparison
// While time retrieved from the application is older than in the database, add the notification
while ($row = mysqli_fetch_assoc($data)) {
  if ($notificationTime < $row['time']) {
    $returnArray[] = array("time"=>$row['time'], "description"=>$row['description']);
  }
}

echo json_encode(array ("entries"=>$returnArray));
```

Figure 19. Send all new notifications into the application after retrieved device id has been controlled

E. Send a table into the application

Based on the request from the application, a certain table will be sent into it. Retrieved device ID is again controlled as well as the time information to avoid duplication of data. Send it either in .csv form or json-encoded based on the request. Code below shows this technique used in one table, but the process remains the same for all tables.

```
// Schedule
else if(isset($_POST['scheduleCheck']) || isset($_GET['scheduleCheck']))  {
  // Select data appropriate for the schedule
  $result = mysqli_query($dbcon, "SELECT lastUpdated FROM times WHERE fileName='schedule'");
  // Fetch $result as associative array
  $row = mysqli_fetch_assoc($result);
  // Retrieve the 'latest upload time' from the database
  $lastTime = $row['lastUpdated'];
  // If the time from the database is newer than from the application
  if ($lastTime > $receivedTime) {
    die("There is nothing new to be updated");
  }

  // Select everything from the chosen table of the database
  $sql = mysqli_query($dbcon, "SELECT * FROM schedule") or die('query failed');

  if (isset($_POST['useJson'])) {
    $returnArray = array();
    while($row = mysqli_fetch_assoc($sql)) {
      $returnArray[] = array (
      'date' => $row['date'],
      'time' => $row['time'],
      'description' => $row['description'],
      );
    echo json_encode(array ("entries"=>$returnArray));
    }
  }
  else {
    // Get the number of rows of the table
    $num_rows = mysqli_num_rows($sql);

    if($num_rows >= 1) {
      // Retrieve the first row of the table as an associative array (headlines)
      $row = mysqli_fetch_assoc($sql);
      // Put empty value into the separators
      $seperator = "";
      $comma = "";
     // Go through the array ($row)
     foreach($row as $name => $value) {
       $separator .= $comma . '' .str_replace('','""', $name);
       $comma = ",";
     }
     // php separator
     $separator .="\n";

     // Do this for the whole .csv file
     while($row = mysqli_fetch_assoc($sql)) {
       $seperator = "";
       $comma = "";
       foreach($row as $name => $value) {
         // Do not override the variable, just add to it the new value
         $separator .= $comma . '' .str_replace('','""', $value);
         $comma = ",";
       }
       // Add a new line in the end
       $separator .="\n";
     }
     // Echo the output
     echo $separator;
    }
  }
}
```

Figure 20. Send table 'schedule' into the application

Approximate Word count: 805