Introduction.

Visual Studio code was used to create a CRUD management system for the client's architecture firm. The languages used are CSS, Javascript, HTML, and PHP. It uses a web hoster, Hostinger to facilitate access to the website and database. The program allows the Client to read, create, filter, update, delete, and manage current instances of employees, subcontractors, and overheads and has the ability to compute certain tasks like estimating costs.

PHP mysqli Library Methods and Their Purposes

mysqli: Is an extension that coders can use to connect and use a MySQL database.

Methods	Purpose
mysqli_connect()	Establishes a connection to the database. Representing it as an object representing the connection to the MySQL server.
$conn = mysqli_connect($servername, $username, $password, $dbname);	
mysqli_query()	Performs a query against the database. For successful queries, it will be represented as a mysqli_result object.
$result = mysqli_query($conn, "SELECT * FROM karyawan WHERE id = '$id'");	
mysqli_close()	Closes the database connection.
mysqli_close($conn);	
mysqli_insert_id()	Retrieves the ID from a query on a table with an AUTO_INCREMENT(AI) column.

if (mysqli_query($conn, $presentsql)) {
$last_id = mysqli_insert_id($conn);
 	
}	

mysqli_error()	Returns a report of the error encountered.

if (!$conn) {
die("Failed db connection: " . mysqli_connect_error());
}	

mysqli_fetch_assoc()	Fetches a result row. It is used to get a row from the result.

while ($row = mysqli_fetch_assoc($result)) {
…
}	

Bootstrap components and their functions

Bootstrap is a frontend toolkit I used to code a lot of my front-end functionality.

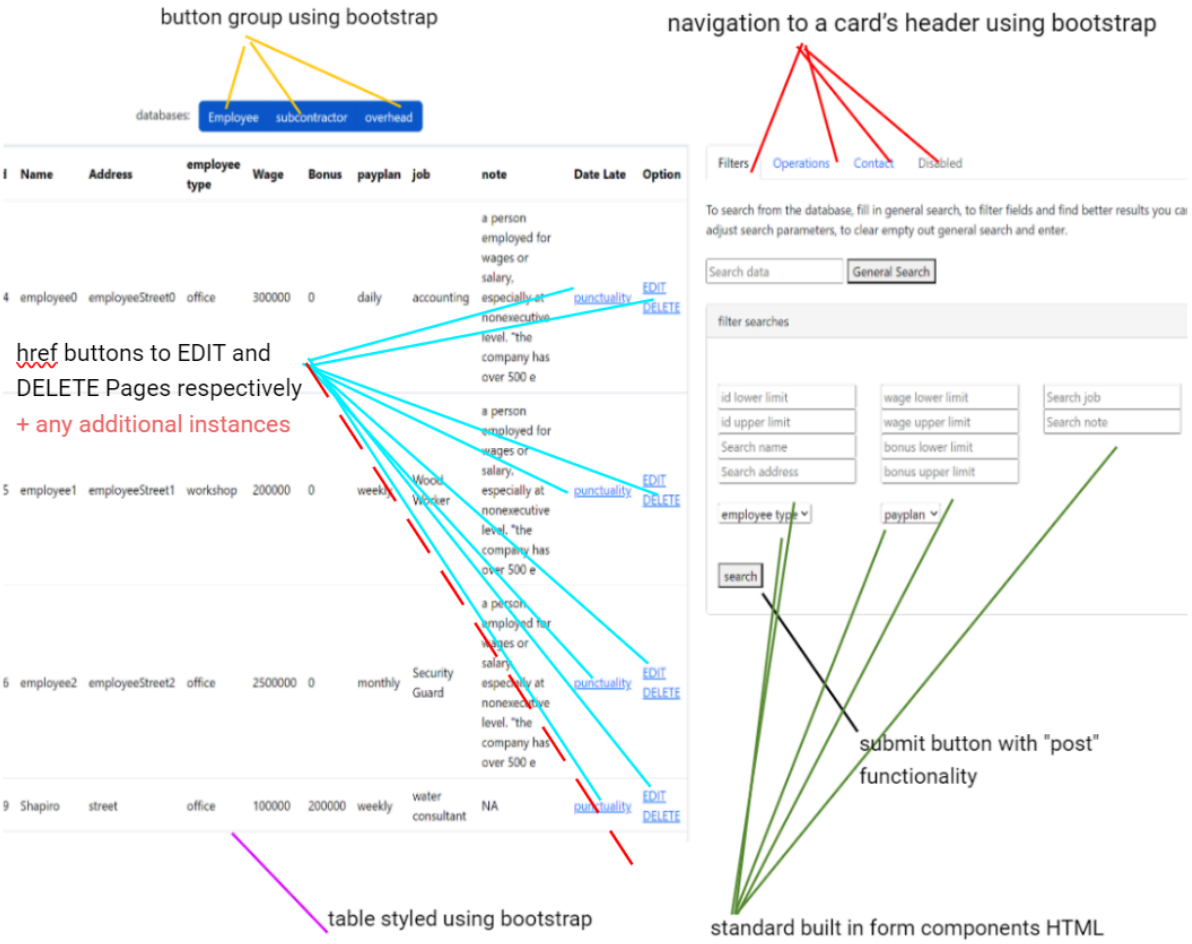
component	Purpose
Navbar	The navbar component is used to create a responsive navigation header for my website or application.
<nav class="navbar navbar-expand-lg navbar-light bg-light">	
Columns	Columns build on the grid's flexbox architecture. Allowing customization of groups of columns, allowing them to grow, shrink, or otherwise change, it sort of acts as a localized divisor(bootstrap, n.d).
Card + Navigation	A card is a flexible and extensible content container, allowing similar localized areas, and with navigation can be divided into pages.
 <h7 class="card-header">filter searches</h7> 	

nav tabs	nav-tabs are used to create tabbable localized pages, even dynamic via JavaScript.
<ul class="nav nav-tabs" id="myTab" role="tablist">	

Techniques used

GUI development

a) **Employee view page, which allows the client to view and manipulate employee instances, along with showing some simple arithmetic operations that can be done on the code.**

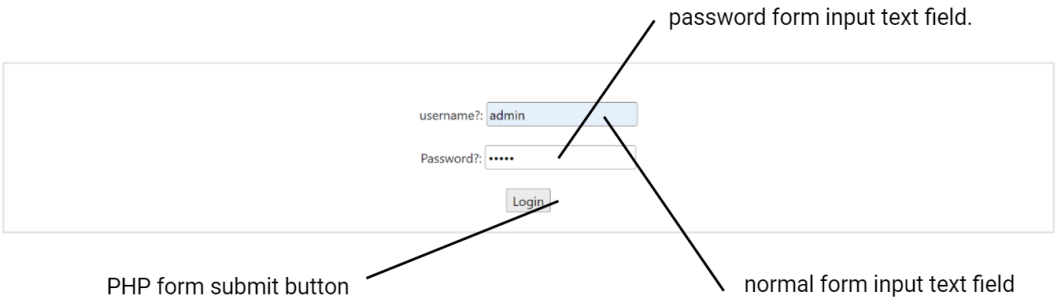


a) **Navigation bar component allows the traversal of the program**

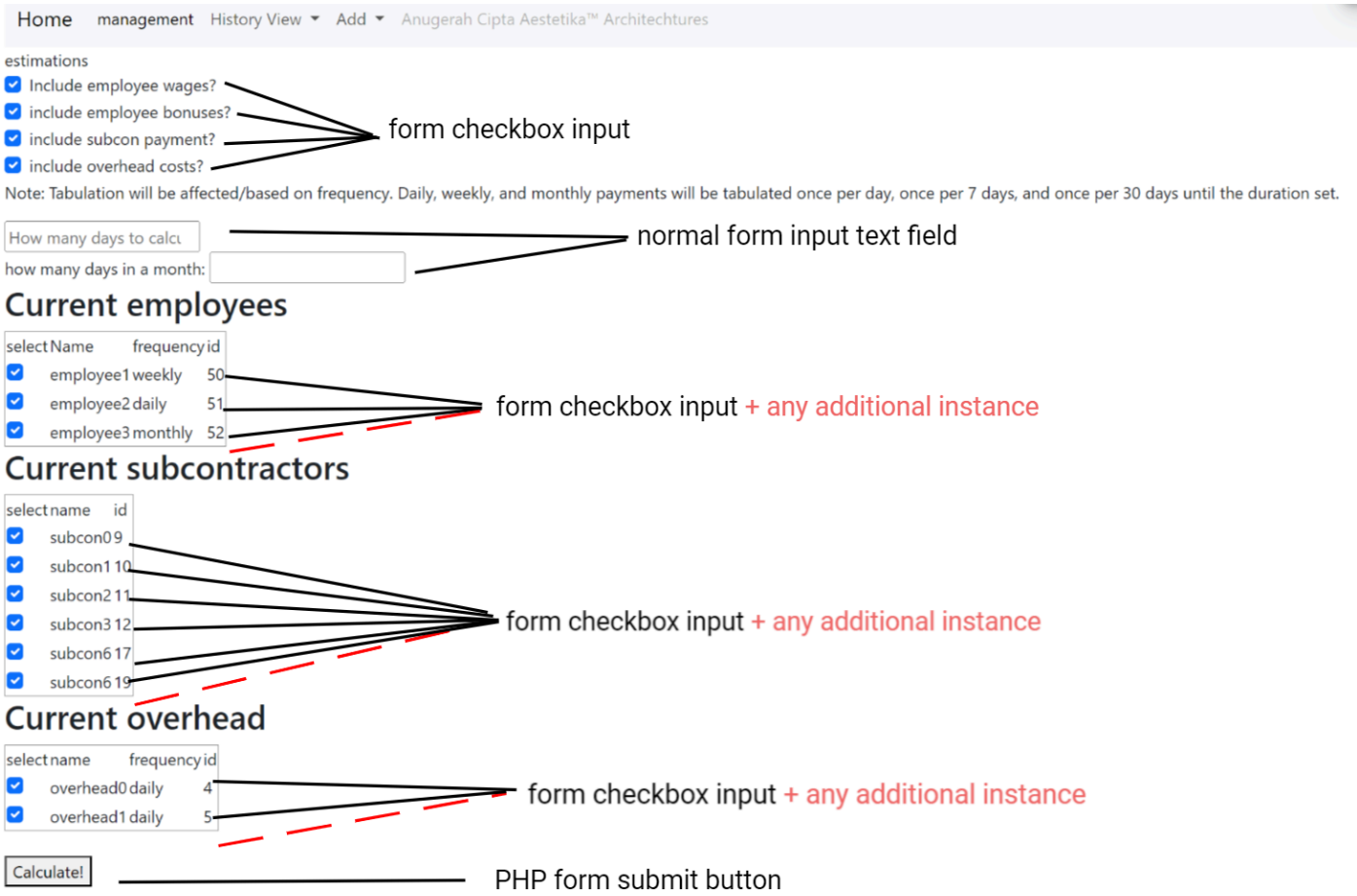
b) **Href allows hyperlinking to different pages for the calendar page, EDIT instance page, and DELETE instance page respectively.**

Date Late	Option
punctuality	EDIT DELETE
punctuality	EDIT DELETE
punctuality	EDIT DELETE

c) **Login page, used as a page for the client to authenticate himself, and restricts unwanted access to the software and data.**

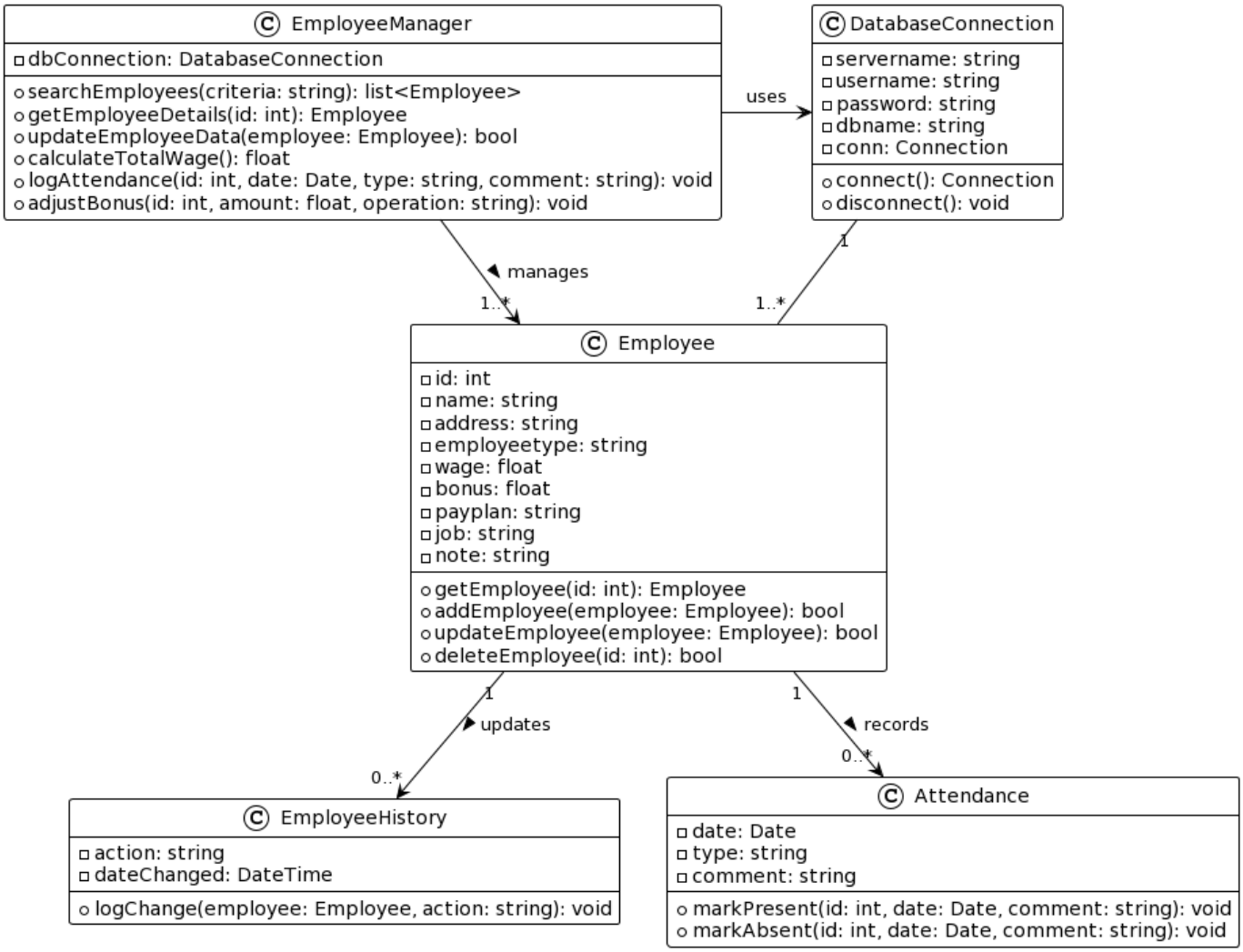


d) Management page, allows for estimation of costs and the such for a period of time



UML Diagrams

UML for the employee project view page



Coding

Database connection:

```php
<?php
    $servername = "localhost";
    $username = "u855808231_root";
    $password = "";
    $dbname = "u855808231_huyo";
$conn = mysqli_connect($servername, $username, $password, $dbname);
if (!$conn) {
    die("error with connection: " . mysqli_connect_error());
}

```

The code above is to establish a connection to the MySQL database hosted on the hosting server. Currently, I have deleted the password which is needed to finish the connection for private purposes. It defines the credentials like server name, username and password, and the database that needs to be connected to. These are done through the mysqli_connect function method, if it is successful, then the $conn variable will hold the connection resource that is used to interact with the database. If it fails, an error message detailing the issue will be outputted to the user.

Nested if Else statement

```php
    if (isset($_POST['presentsubmit'])) {
        $id = $_GET['id'];
        $present = $_POST['presentdate'];
        $comment = $_POST['comment'];
        $datecreated = date("Y-m-d H:i:s");


        $sql = "SELECT * FROM calander WHERE (presentdate='$present' OR absentdate ='$present') AND id='$id'";
        $result = mysqli_query($conn, $sql);
        if (mysqli_num_rows($result) > 0) {
            echo "<script>alert('There has already been a date set for this date, please delete old to input new!');</script>";
        } else {
            $presentsql = "INSERT INTO calander (id,presentdate,datecreated,comments) VALUES ('$id','$present','$datecreated','$comment')";
            if (mysqli_query($conn, $presentsql)) {
                $last_id = mysqli_insert_id($conn);

        }
    }
}
```

The code above is a snippet of my code responsible for form submission in my employee database's calendar/punctuality page. It handles the form submission for when the user inputs in the submission of present dates, a similar one is seen for absent dates if the absent submit is pressed instead. The if statement checks whether the presentsubmit button POST variable is set, which means a form has been submitted. Once the POST variable is set into variables to be used, the $sql method uses the SELECT functionality to check whether there is already a date for the inputted present date. This query is stored in another variable, $result along with the connection method and if there is a matching entry, (mysqli_num_rows($result) > 0) then an error echo is presented to the user. Else, the data is inserted into the calendar database. If the insert query is successful, it retrieves the last inserted ID and stores it in the $last_id variable.

Filter handling:

```php
elseif (isset($_POST['submit'])) {
  $typesearch = $_POST['typesearch'];
  $namesearch = $_POST['namesearch'];
  $payplansearch = $_POST['payplansearch'];
  $jobsearch = $_POST['jobsearch'];
  $notesearch = $_POST['notesearch'];
  $addresssearch = $_POST['addresssearch'];

  $selectidquery = "SELECT * FROM karyawan";
  $startidsearch = $_POST['startidsearch'] ?? null;
  $endidsearch = $_POST['endidsearch'] ?? null;

  $selectwagequery = "SELECT * FROM karyawan";
  $startwagesearch = $_POST['startwagesearch'] ?? null;
  $endwagesearch = $_POST['endwagesearch'] ?? null;

  $selectbonusquery = "SELECT * FROM karyawan";
  $startbonussearch = $_POST['startbonussearch'] ?? null;
  $endbonussearch = $_POST['endbonussearch'] ?? null;

  $minimumandmaximum = "SELECT MIN(id) AS minid, MAX(id) AS maxid FROM karyawan";
  $minandmaxwage = "SELECT MIN(gaji) AS minwage, MAX(gaji) AS maxwage FROM karyawan";
  $minandmaxbonus = "SELECT MIN(bonus) AS minbonus, MAX(bonus) AS maxbonus FROM karyawan";

  $idresult = mysqli_query($conn, $minimumandmaximum);
  $defaultrow = mysqli_fetch_assoc($idresult);

  $wageresult = mysqli_query($conn, $minandmaxwage);
  $defaultwagerow = mysqli_fetch_assoc($wageresult);

  $bonusresult = mysqli_query($conn, $minandmaxbonus);
  $defaultbonusrow = mysqli_fetch_assoc($bonusresult);
  if (empty($startidsearch)) {
      $startidsearch = $defaultrow['minid'];
  }
  if (empty($endidsearch)) {
    $endidsearch = $defaultrow['maxid'];
  }
  if (empty($startwagesearch)) {
    $startwagesearch = $defaultwagerow['minwage'];
  }
  if (empty($endwagesearch)) {
  $endwagesearch = $defaultwagerow['maxwage'];
  }
```

```php
  if (empty($startbonussearch)) {
    $startbonussearch = $defaultbonusrow['minbonus'];
  }
  if (empty($endbonussearch)) {
  $endbonussearch = $defaultbonusrow['maxbonus'];
  }
  #### TAB THIS to make pretty
  $sumqueryfilter = "SELECT SUM(gaji) AS totalwage FROM karyawan WHERE employeetype LIKE '%$typesearch%' AND Nama LIKE '%$namesearch%' AND id BETWEEN $startidsearch AND $endidsearch AND al
  $sumbonusfilter = "SELECT SUM(bonus) AS totalbonus FROM karyawan WHERE employeetype LIKE '%$typesearch%' AND Nama LIKE '%$namesearch%' AND id BETWEEN $startidsearch AND $endidsearch AND
  $sumresult = mysqli_query($conn, $sumqueryfilter);
  $sumbonus = mysqli_query($conn, $sumbonusfilter);
  if ($sumresult) {
    ?>
      <br>
      <?php
    $row = mysqli_fetch_assoc($sumresult);
    $totalwage = $row['totalwage'];
    $formattedtotalwage = number_format($totalwage, 0, '.', ',');
    $brow = mysqli_fetch_assoc($sumbonus);
    $totalbonus = $brow['totalbonus'];
    $formattedtotalbonus = number_format($totalbonus, 0, '.', ',');
    echo "Total Wage: " . $formattedtotalwage;
    $totalwagebonus1 = $totalwage + $totalbonus;
    $formattedtotalwagebonus = number_format($totalwagebonus1, 0, '.', ',');
    ?>
    <br>
    <?php
      echo "Total Wage + Bonus: " . $formattedtotalwagebonus;
  }
```

The code snippet above handles when the submit post variable is set through the submit button form button where a search form is submitted to filter and aggregate data from the employee database. The inputs are search parameters from a filter form submitted by the user and are stored in specific variables. The snippet also has several SQL queries to fetch min and max values of numerical fields like id, and wage where if null, a default parameter is set before the end at the max and min ranges to allow all data to be shown. The script also constructs two SQL queries using the user's search parameters and the defaults used to calculate the sum of some of the numerical values of wage and bonus that fit the filter, where they are then formatted into a specific string format and also increment by monetary separations like 1,000,000.

Table output

```
elseif (isset($_POST['submit'])) {
  if ($queryresults) {
  if(mysqli_num_rows($queryresults)>0){

    while ($row = mysqli_fetch_assoc($queryresults)) {
      echo '<tr>';
      echo '<td>'.$row['id'].'</td>';
      echo '<td>'.$row['Nama'].'</td>';
      echo '<td>'.$row['alamat'].'</td>';
      echo '<td>'.$row['employeetype'].'</td>';
      echo '<td>'.$row['gaji'].'</td>';
      echo '<td>'.$row['bonus'].'</td>';
      echo '<td>'.$row['payplan'].'</td>';
      echo '<td>'.$row['job'].'</td>';
      echo '<td>'.$row['note'].'</td>';
      echo "<td> <a href='calander.php?id=" . $row['id'] . "'>punctuality</a></td>";
      echo "<td> <a href='employee/edit.php?id=" . $row['id'] . "'>EDIT</a> <a href='employee/delete.php?id=" . $row['id'] . "'>DELETE</a></td>";
      echo '</tr>';
    }
  } else {
    echo '<tr>';
    echo '<td>no data found</td>';
    echo '</tr>';
    }
  }
}
```

The snippet above is responsible for displaying the table with the filtered results from the submit text field. If the submit post variable is set, showing that the form is submitted, then the code snippet executes. The query results are part of a variable that uses the submit search parameters in $filterquery to filter search results outputted to only the ones that have been filtered for in the form ($filterquery = "SELECT * FROM employee WHERE employeetype LIKE '%$typesearch%' ….); If there are results for these parameters, it enters a loop that fetches all instances (mysqli_fetch_assoc($queryresults)) and iterates over each row of the result set displaying the row value in the database as a table for the user. If no instances meet the search criteria, then a “no data found” is displayed.

Session check

```
1  <?php
2  session_start();
3
4  if (!isset($_SESSION['loggedin']) || $_SESSION['loggedin'] !== true) {
5      echo "<script>alert('This path is restricted go back to index.php!');</script>";
6      exit;
7  }
8  ?>
9
```

The code snippet is a security check to ensure that a user trying to access the page has been authorized by checking if he has gone through the login page. This is placed at the top of the file ot restrict access to authenticated users only. The session start starts or continues an existing session created by the user. Used to maintain a state across multiple pages, used in keeping track of if the user is loggin in. It checks $_SESSION for a loggedin variable declared in the login page if the user inputs the correct username and password and loggedin is set as a TRUE boolean value. If they are not logged in, they are prompted to go back to the index page and log in.

Updating instances

```
31
32      $insertquery = "INSERT INTO subconhistory (historyaction, historyname, historywork, historypay, leftpay, historyid, datechanged2) VALUES ('$action','$historyname','
33      if (mysqli_query($conn, $insertquery)) {
34          echo "Data inserted into new table successfully.";
35      } else {
36          echo "Error inserting data into new table: " . mysqli_error($conn);
37      }
38
39      $name = $_POST['Newname'];
40      $work = $_POST['Newwork'];
41      $pay = $_POST['newPay'];
42      $id = $_POST['id'];
43
44      $updatequery = "UPDATE subcon SET name = '$name', work = '$work', pay = '$pay', id = '$id' WHERE id = '$id'";
45
46      if (mysqli_query($conn, $updatequery)) {
47          echo "Success.";
48      } else {
49          echo "Error: " . mysqli_error($conn);
50      }
51      $updatelefttopay = "UPDATE subcon SET newPay = '$pay' WHERE id = '$id'";
52      if(mysqli_query($conn, $updatelefttopay)){
53          echo "updated successfully.";
54          header('Location: /IA/subconprojectview.php');
55      } else {
56          echo "Error updating left to pay: " . mysqli_error($conn);
57      }
```

This code snippet is part of the update subcontractors.php page. The code checks for the ID clicked on using the EDIT href attribute in the employee view page meant to post the ID value of the instance clicked to the edit.php field page before being posted to the update.php page (this page). If there is a record found for the ID, then some variables are declared like the current time and some of the variables are declared from edit.php, the $action variable is the string “edit”. The previous values are then stored in the Subcon history, along with the date of changing and action, before the new values declared in the form are used to update the Subcon table for that instance where id = $id

Word count: 961 (excluding tables)

References

1. *Autofill form fields based on value from select field*. Stack Overflow. (2016, August 1). https://stackoverflow.com/questions/32803251/autofill-form-fields-based-on-value-from-select-field
2. Hadi, D. A. (2016, February 13). *Membuat Crud Dengan PHP dan mysql edit data*. Malas Ngoding. https://www.malasngoding.com/membuat-crud-dengan-php-dan-mysql-edit-data/
3. Hadi, D. A. (2018, March 15). *Membuat Crud Dengan PHP dan MySQLi - Hapus Data*. Malas Ngoding. https://www.malasngoding.com/membuat-crud-dengan-php-dan-mysqli-hapus-data/
4. *HTML* . HTML input type="datetime-local". (n.d.). https://www.w3schools.com/tags/att_input_type_datetime-local.asp#:~:text=Definition%20and%20Usage,tag%20for%20best%20accessibility%20practices
5. *How to pop an alert message box using php?*. Stack Overflow. (2011, November 1). https://stackoverflow.com/questions/13851528/how-to-pop-an-alert-message-box-using-php
6. *How to add extra whitespace in php?*. Stack Overflow. (2000, January 1). https://stackoverflow.com/questions/2300142/how-to-add-extra-whitespace-in-php
7. Olawanletjoel, J. (2023, April 14). *Javascript get current date – today's date in Js*. freeCodeCamp.org. https://www.freecodecamp.org/news/javascript-get-current-date-todays-date-in-js/
8. *PHP header(Location: ...): Force URL change in address bar*. Stack Overflow. (2012, August 1). https://stackoverflow.com/questions/7467330/php-headerlocation-force-url-change-in-address-bar
9. PHP number_format() function. (n.d.). https://www.w3schools.com/php/func_string_number_format.asp
10. YouTube. (2021, August 15). *#12. search data using PHP and mysql database.* YouTube. https://www.youtube.com/watch?v=9ANd4KVPQtE&ab_channel=StepbyStep

Bootstrap elements used in code:

1. Mark Otto, J. T. (n.d.). *Containers*. · Bootstrap v5.3. https://getbootstrap.com/docs/5.3/layout/containers/
2. Mark Otto, J. T. (n.d.-b). *Grid system*. · Bootstrap v5.3. https://getbootstrap.com/docs/5.3/layout/grid/
3. Mark Otto, J. T. (n.d.-c). *Tables*. · Bootstrap v5.3. https://getbootstrap.com/docs/5.3/content/tables/
4. Mark Otto, J. T. (n.d.-a). *Checks and radios*. · Bootstrap v5.3. https://getbootstrap.com/docs/5.3/forms/checks-radios/

5. Mark Otto, J. T. (n.d.-a). *Button Group*. · Bootstrap v5.3. https://getbootstrap.com/docs/5.3/components/button-group/
6. Mark Otto, J. T. (n.d.-e). *Navs and tabs*. · Bootstrap v5.3. https://getbootstrap.com/docs/5.3/components/navs-tabs/
7. Mark Otto, J. T. (n.d.-e). *Navbar*. · Bootstrap v5.3. https://getbootstrap.com/docs/5.3/components/navbar/