```php
<?php

use mikehaertl\wkhtmlto\Pdf;

/* Database connection settings */
$host = "omitted";
$user = "admin";
$pass = "omitted";
$db = "innodb";
$mysqli = new mysqli($host,$user,$pass,$db) or die($mysqli->error);



function check_login() {
    //  Allows errors to display on all pages
    error_reporting(E_ALL);
    error_reporting(-1);
    ini_set('display_errors', true);

    //  Checks login
    if ( $_SESSION['logged_in'] != true ) {
        if ($_SESSION['attempt']) {
            $_SESSION['rmessage'] = "Confirmation link has been sent. Please
verify your account by clicking on the link in the message!";
            $_SESSION['goodbad'] = 'good';

        }
        else {
          $_SESSION['rmessage'] = "You must log in before viewing your
profile page";
          $_SESSION['goodbad'] = 'bad';
        }
        header("location: ../accounts/registration.php");
    }
    else {
        $_SESSION['rmessage'] = '';
        $_SESSION['goodbad'] = 'good';
    }
}


function safe_query($mysqli, $query_string) {
    $result = $mysqli -> query($query_string);
    if(! empty( $mysqli->error ) ){
        console_log($query_string);
        console_log($mysqli->error);
    }
//     console_log('safe query'.$query_string);
    return $result;
}


function make_equality_string($eq_dict, $conj=' and ') {
    $keys = array_keys($eq_dict);
    $equality_string = '';
    foreach($keys as $key) {
        if ($equality_string == '') {
```

```
            $conjunction = '';
        }
        else {
            $conjunction = $conj;
            }
        if ($eq_dict[$key] == null) {
            $equality_string = $equality_string.$conjunction.$key." IS null
";
        }
        else {
            $equality_string = $equality_string.$conjunction.$key." =
'".$eq_dict[$key]."'";

        }
    }
    return $equality_string;
}


function dict_sanitize($mysqli, $dict) {
    foreach ($dict as $key => $element) {
        $element = $mysqli->escape_string($element);
        $element = addcslashes($element, '%_');
        $dict[$key] = $element;
    }

    return $dict;
}



function select_query($mysqli, $table_name, $eq_dict, $desc_field=Null) {
    # make the query
    $eq_dict = dict_sanitize($mysqli, $eq_dict);


    $equality_string = make_equality_string($eq_dict);
    $qp1 = "select * from ".$table_name." where ".$equality_string;
    $qp2 = ";";

    if ($desc_field != NULL) {
        $qp2 = " ORDER BY ".$desc_field." DESC;";
    }

    $query = $qp1.$qp2;

    $output = safe_query($mysqli, $query);
    console_log($query);
    console_log("Select Query Executed --> Num rows: ".$output->num_rows);
    return $output;
}


function insert_query($mysqli, $table_name, $insert_dict) {
    $insert_dict = dict_sanitize($mysqli, $insert_dict);

    $fields = array_keys($insert_dict);
```

```
    $values = $insert_dict;
    $f_string = implode(", ", $fields);
    $v_string = '';


    foreach ($values as $value) {
        if ($value == null) {
            $v_string = $v_string." null,";
        }
        else {
            $v_string = $v_string."'".$value."',";
        }
    }
//     $v_string = str_replace(',);', ');', $v_string);

    $v_string = rtrim($v_string, ',);');

    $query = "insert into ".$table_name." (".$f_string.") VALUES
(".$v_string.");";
    console_log($query);
    return safe_query($mysqli, $query);
//     if wait == False:
//         return make_query(query)
//     else:
//         global wait_string
//         wait_string = wait_string + query

}


function update_query($mysqli, $table_name, $eq_dict, $change_dict) {
    $eq_dict = dict_sanitize($mysqli, $eq_dict);
    $change_dict = dict_sanitize($mysqli, $change_dict);


    $equality_string = make_equality_string($eq_dict);
    $change_string = make_equality_string($change_dict, $conj=', ');
    $query = 'UPDATE '.$table_name.' SET '.$change_string.' WHERE
'.$equality_string.';';
    console_log($query);
    return safe_query($mysqli, $query);
}


function console_log($message, $priority=1) {
    $curr_priority = 1;
    if ($priority <= $curr_priority) {
    $STDERR = fopen("php://stderr", "w");
              fwrite($STDERR, "\n".$message."\n\n");
              fclose($STDERR);
    }
}


// Post, Redirect, Get --> stops from resubmission on page refresh
function prg() {
    if ($_SERVER['REQUEST_METHOD'] == 'POST') {
```

```
        header("Location: {$_SERVER['REQUEST_URI']}", true, 303);
    }
}


function make_time($kids) {
    $num_pages = 0;

    if ($kids == 1) {
        $wait = 1444;
        return $wait;
    }

    foreach ($kids as $kid) {
        if ($kid['level'] == 0) {
            $num_pages += 16;
        }
        else {
            $num_pages += 4;
        }
    }

    console_log("Nummpagess:".$num_pages);
    $wait = 1000 + ($num_pages * 111);
    return $wait;
}



function make_pdf($html_string, $kids) {
    require_once '../vendor/autoload.php';

    $wait = make_time($kids);

    console_log("Wait time:".$wait);
    $options = array('javascript-delay' => $wait, 'margin-top' => '50mm');
    $pdf = new Pdf;
    $pdf->setOptions($options);
    $pdf->addPage($html_string);

    if (!$pdf->send('wksht_test.pdf')) {
        console_log('Error: pdf not sent');
        $error = $pdf->getError();
        console_log($error);
    }
}


function echo_json($array) {
    $myJSON = json_encode($array);

    echo $myJSON;
}


function python_slice($string, $start, $end) {
    $string = substr($string, $start, (strlen($string) - $start));
```

```
}


function unique_multidim_array($array, $key) {
    $temp_array = array();
    $i = 0;
    $key_array = array();

    foreach($array as $val) {
        if (!in_array($val[$key], $key_array)) {
            $key_array[$i] = $val[$key];
            $temp_array[$i] = $val;
        }
        $i++;
    }
    return $temp_array;
}


function e($string) {
    $ciphering = "AES-256-CTR";
    $encryption_iv = '1234567891011121';
    $encryption_key = "UCkey";
    $options = 0;

    $encryption = openssl_encrypt($string, $ciphering, $encryption_key,
$options, $encryption_iv);
    $encryption = urlencode($encryption);

    return $encryption;
}


function d($encryption) {
    $ciphering = "AES-256-CTR";
    $decryption_iv = '1234567891011121';
    $decryption_key = "UCkey";
    $options = 0;

    $encryption = urldecode($encryption);
    $decryption = openssl_decrypt($encryption, $ciphering, $decryption_key,
$options, $decryption_iv);

    return $decryption;
}


?>
```