CSCI 3342/6312 – Tutorial/Assignment #4 **Deadline: 3:00 pm on June 12, 2020**

– Quiz #4 (Separate *Container*) **Deadline: 3:00 pm on June 12, 2020**

Completion: Required **Submission: Required**

Last Name:_____ First Name:_____

This *tutorial* is to help you learn MySQL relational database and a cloud platform to host the database on. Once, the database is hosted, the tutorial helps you to learn uploading the values of sensors (interfaced with your Raspberry Pi) to the database, and to synchronize the actuator connected to RPi according to the database.

1) How and what to submit?

Submit the following (upload in Blackboard to the available container) in "one" PDF document (not in docx or any other format):

i) The certification page (see next page) should be the first page, followed by
ii) your solution to the problems given on this assignment.

One way is to copy the certification page and your solution to the problems into a Word document and then save the Word document as PDF, and upload the PDF version (not docx version). Only the PDF version will be graded.

2) Only ONE upload attempt is allowed: Before submitting a document through Blackboard, you should review the document being uploaded to make sure that you are uploading the correct document (e.g. do not upload the assignment belonging to another course). To help you prevent uploading wrong documents, notes (titled "HelpOnSubmissionThroughBlackboard" on how to save & review drafts before final submission have been uploaded under Reference Material folder.

Certification Page

This page must be the first page of your uploaded document.
Your assignment will not be graded without this page (completed with your full name in the area provided) as the first page of your uploaded document.

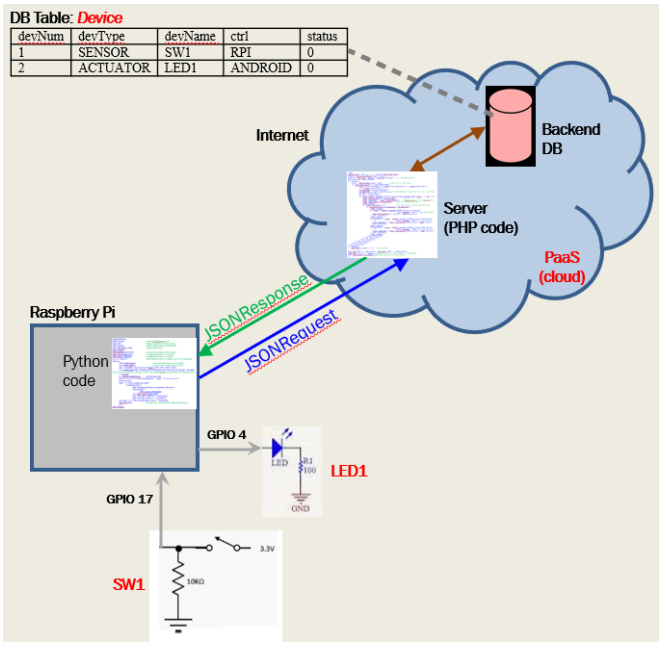
I, _____, certify that the work I am uploading represents my own efforts, and is not copied from anyone else or any other resource (such as Internet). *Furthermore, I certify that I have not let anyone copy from my work.*

Tutorial Portion

Learning Objectives!

There are several cloud services available: **IaaS** (*Infrastructure as a Service*... where one can get a server with ***n*** number of CPU cores, ***k*** GB of RAM, and storage to install one's software), **SaaS** (*Software as a Service*... where one can subscribe to use a software, e.g. Office365), and **PaaS** (*Platform as a Service*... where one can use mixed resources to develop an application).

In this tutorial we will be using a PaaS in the cloud (www.000webhost.com) to do the following tasks as depicted in the diagram on the right:



a) Install DBOR (Database of Record) in PaaS to maintain a database table called device to keep track of the status of **SW1** (simulating a sensor) and **LED1** (simulating an actuator) interfaced with a Raspberry Pi.

b) Use JSONRequest (embedded in Python) to send the status of SW1 to the DBOR in the cloud.

c) Use JSONResponse (embedded in PHP) in the cloud PaaS to send the status of LED1 to Raspberry Pi.

Below are some youtube videos for getting familiar with the topics covered in this tutorial:

PHP ➔ https://www.youtube.com/watch?v=7TF00hJI78Y
PHP MySQL ➔https://www.youtube.com/watch?v=mpQts3ezPVg
Protecting DB from SQL Injection ➔ https://www.youtube.com/watch?v=nTgFPcYRkys
mysqli prepare statements ➔ https://www.youtube.com/watch?v=I4JYwRIjX6c

☞ If you are not familiar with the basic MySQL database functionality, please, consider completing a tutorial on MySQL, ***Tutorial_Intro_to_MySQL***, available under ***Reference Material*** folder on Blackboard.

Below is a summary of the sections in this tutorial:

Section 0: Create an account on a PaaS, and setup the environment to meet our needs.

Section 1: Create DBOR on PaaS.

Section 2: Populate DBOR.

Section 3: Provide DBOR connection information in PaaS.

Section 4: Restrict the *form requests* to POST type.

Section 5: Register/setup users with usernames and *hashed* passwords.

Section 6: *Setup & test* Python code on RPi to interact with the PHP code of Section #7 (on PaaS) to power on or off LED1 from the cloud, and to update the status of SW1 in the cloud according to RPi.

Section 7: PHP code on PaaS to interact with the code of Section #6.

0) Creating an account on PaaS, enabling FTP, and redirecting HTTP to HTTPS

a) Creating an account on a server site: Access ***www.000webhost.com***, and create an account using one of your existing email addresses *(such as your gmail, yahoo, or UTRGV email address)* by clicking on ***Free Sign Up***. Select *non-pro* option when asked. You will be sent a request for confirming your email address that you provided as the username. You are required to confirm your email address before using the site. *Do not use the same password that you have for your existing email address since you may have to use this password in plain text in your program.*

b) Enabling FTP on the server site: Log into *000webhost.com* site with your credentials for the site, and access ***Settings*** ➔ ***General***. Under FTP details, make sure that FTP transfer is ***ON***. Make note of the host name, port, username, and password. This information is needed to transfer a file from RPi to this server using FTP. ***(In production environment, FTP must not be used. Instead SFTP – Secured FTP - must be used, however, this site does not offer SFTP).***

FTP transfer	ON
Host Name:	files.000webhost.com
Port:	21
Username	janedoe
Password:	same as your website password

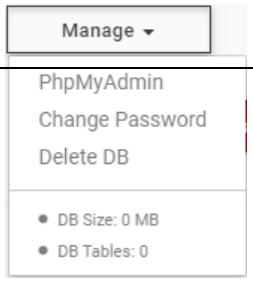
c) Redirecting webpages from HTTP to HTTPS on the server site: Access ***File manager*** ➔ ***Upload files now*** on ***www.000webhost.com*** (it may take a few minutes before contents are displayed). Edit ***htaccess*** file (right click on ***htaccess*** and select ***Edit***… *this is a configuration file used on Apache web servers*) to include the following code so that webpages from HTTP are redirected to HTTPS since ***000webhost.com*** provides HTTPS by default:

```
RewriteEngine  On
RewriteCond  %{HTTPS}  off
RewriteRule  ^(.*)$  https://%{REQUEST_URI}  [L,R=301]
```

1) Creating DBOR on the server site: Access *000webhost.com* using your credentials. Click on ***Manage database*** icon to access ***Manage Databases*** site, followed by clicking on ***New Database*** link. In the Create new database dialog box, provide ***Database name***, ***Database username*** and ***Password***:

Database name: insurance
Database username: user
Password: mypass

It may take a few minutes to create the database. You may see the following as a response to your request for creating the database:

DB Name	DB User	DB Host	
id6147251_insurance	id6147251_user	localhost	

Make note of the options available under the drop-down ***Manage*** menu.

2) Populating DBOR: Under the ***Manage Databases*** site, click on the drop-down ***Manage*** menu and select ***PhpMyAdmin*** option. Enter DB user (e.g. ***id6147251_user***) and the corresponding password. Click on the ***Databases*** tab and make sure that insurance database (e.g. ***id6147251_insurance***) is selected, otherwise, you will get *access denied* type of errors when inserting tables. Select the ***SQL*** tab and insert SQL queries for creating tables and populating them. After inserting the SQL queries/statements given below, click on ***Go*** to run the SQL queries to create tables and populate them:

create table person (pname VARCHAR(30) NOT NULL, street VARCHAR(50) NOT NULL, city VARCHAR(50) NOT NULL, Primary Key (pname));
create table vehicle (year VARCHAR(4) NOT NULL, make VARCHAR(30) NOT NULL, model VARCHAR(30) NOT NULL, cost INT NOT NULL, licplate VARCHAR(7) NOT NULL, pname VARCHAR(30) NOT NULL, Primary Key (licplate,pname));
create table accident (accnum INT NOT NULL, licplate VARCHAR(7) NOT NULL, accdate date NOT NULL, pname VARCHAR(30) NOT NULL, Primary Key (accnum,licplate));
insert into person values ("marisol", "zenith", "harlingen"), ("dolly", "pstreet", "brownsville");
insert into vehicle values (2005, 'toyota', 'camry', 25000, 't123', 'marisol');
insert into vehicle values (1996, 'jeep', 'wrangler', 23000, 'm123', 'marisol');
insert into vehicle values (1997, 'suzuki', 'samurai', 25000, 'k123', 'dolly');
insert into accident values (101, 'n123', '2012/07/15', 'sunny'), (102, 'h123', '2014/04/04', 'sunny');

You may want to uncheck the ***Enable foreign key checks*** option if it causes issues (it was left alone while testing this tutorial). You may click on the leftmost tab, which may be named now as ***Structure***, to see the structure of the ***insurance*** database after you executed the SQL queries. You may click on a table to check its contents.

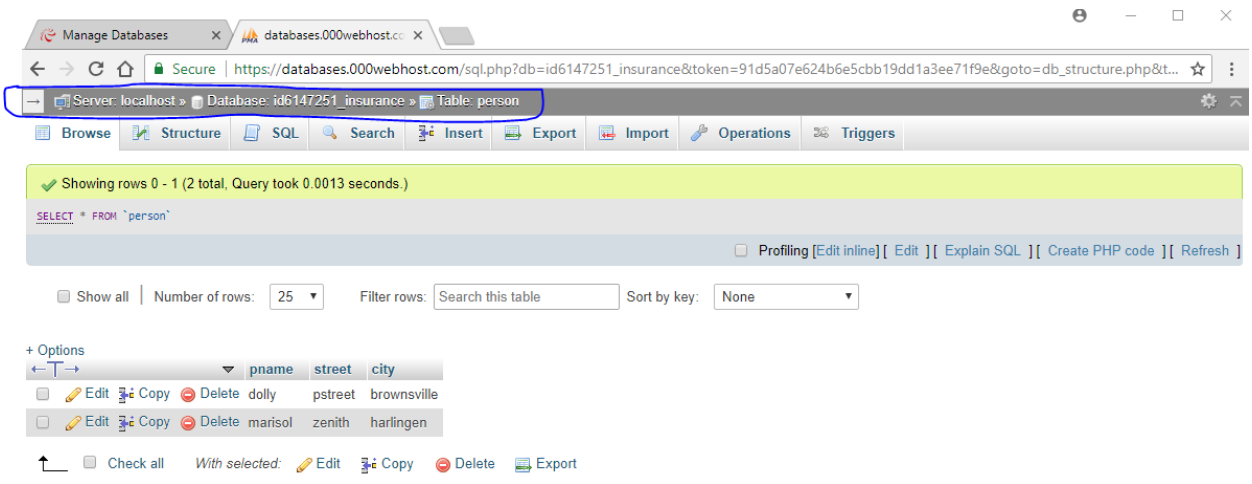
Using the SQL statement give below, create a table called ***webuser*** which will have a list of users (with passwords) to securely access the website that you will develop in an upcoming tutorial.

create table webuser (pname VARCHAR(30) NOT NULL, password VARCHAR(255) NOT NULL, Primary Key (pname));

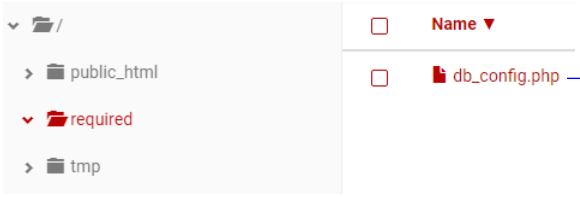
Using the SQL statement give below, create a table called ***device*** which will have a list of devices connected to the RPi:

create table device (devnum INT NOT NULL, devtype VARCHAR(10) NOT NULL, devname VARCHAR(10) NOT NULL, ctrl VARCHAR(10) NOT NULL, status INT NOT NULL, Primary Key (devnum));
insert into device values (1, 'SENSOR', 'SW1', 'RPI', 0);
insert into device values (2, 'ACTUATOR', 'LED1', 'ANDROID', 0);

Make note of the ***breadcrumbs*** (*encircled in blue in the following snapshot*) which may be used to navigate within the page:

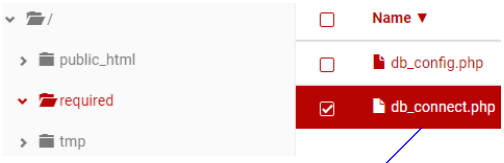


3) Connection information for DBOR: Access ***File manager*** ➔ ***Upload files now*** and create folder named ***required*** outside ***public_html*** folder. Create the php script shown below in the file named ***db_config.php*** (*using your user, password, & DB credentials*) inside the folder ***required***:




```
<?php
define('DB_HOST', 'localhost');
define('DB_USER', 'id6147251_user');
define('DB_PASSWORD', 'mypass');
define('DB_DATABASE', 'id6147251_insurance');
?>
```

Create a file named ***db_connect.php***, with the php script shown below, inside the folder ***required***. This will be used by other php files whenever connection to the database is required:




```
<?php
require_once 'db_config.php';  //tell the script that file is in this folder (not in this script)
$mysqli = new mysqli(DB_HOST, DB_USER, DB_PASSWORD, DB_DATABASE);
?>
```

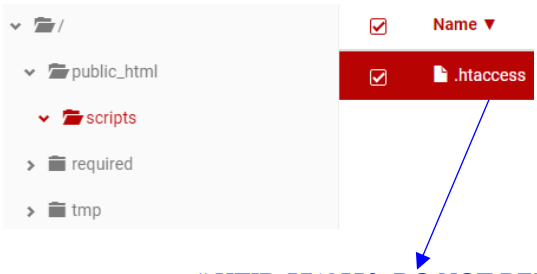
MySQLi is **i**mproved MySQL API used by PHP to interact with MySQL database.

The reason for having these files in a folder outside ***public_html*** is for security reasons so that these files are not accessible by everyone.

4) Allowing only post requests: A form within HTML can use either ***post*** method or ***get*** method to pass data from the client (browser) to the server for processing (e.g. storing the values entered into an HTML form by a user into DBOR). For security reasons, the requests need to be limited to ***post*** method. Create a folder called ***scripts*** inside ***public_html*** folder. Copy the existing ***.htaccess*** file into ***scripts*** folder and add the following lines to the copied ***.htaccess*** file in order to deny ***get*** type of requests:

```
<Limit GET>
deny from all
</Limit>
```

Below is the view after completing the above:




```
# HTID:5712550: DO NOT REMOVE OR MODIFY THIS LINE AND THE LINES BELOW
php_value display_errors 1
# DO NOT REMOVE OR MODIFY THIS LINE AND THE LINES ABOVE HTID:5712550:
RewriteEngine On
RewriteCond %{HTTPS} off
RewriteRule ^(.*)$ https://%{REQUEST_URI} [L,R=301]
<Limit GET>
deny from all
</Limit>
```

5) Registering web users to access your database (or website) securely: It is recommended to hash a password before storing it, rather than storing it in plaintext. A table called ***webuser*** should have been created earlier to have records of users allowed to access your website. Below is a php script to insert a web user's username and hashed password into ***webuser*** table. Store the script in ***public_html➔quick_register.php***. Run the script (right click on ***quick_register.php*** and select ***View***) for each user that you would like to register. When all the web users have been registered, make sure to set $password to "******" in the script so that the actual password of the user is not visible in the script.

```
<?php
require_once __DIR__ . '/../required/db_connect.php';
$username = "ben";
$password = "benpass";            //set this to ****** when done so the script does not show the password
if ($stmt = $mysqli->prepare("INSERT INTO webuser (pname, password) VALUES (?, ?)")) {
   $hashedPassword = password_hash($password, PASSWORD_DEFAULT);
   $stmt->bind_param('ss',$username, $hashedPassword);   //'ss': each s corresponds to the type of param
   $stmt->execute();                                     //First & 2nd params are string type
}
?>
```

mysqli->prepare(sql statement) prevents SQL injection by preventing malicious input. Note that $stmt is an object.

pname	password
ben	hash_value_of_the_password_of_ben
mary	hash_value_of_the_password_of_mary

Check the database table ***webuser*** for users that you registered. The passwords should be represented by their hashed values.

6) Python program on RPi to interact with the server: The php script (*shown in the next section*), ***sync_rpi_data.php***, running on the server is triggered by ***JSONRequest***, shown below, from RPi. Specifically, the following interaction happens:

> ***JSONRequest**, sent to the server every **delay** seconds, carries the status of LED1 and SW1. RPi expects the server to set the status of SW1 in table **device** as it appears in the request. In response, the server makes the changes in DBOR, and sends back the status of LED1 and SW1 as they appear in table **device** (after making the changes). RPi sets LED1 according to the status of LED1 sent by the server in response to **JSONRequest**.*

Write the following program on RPi:

#nano p2_t8.py

```
#!/usr/bin/python
import requests                         #import JSONRequests library
import time                             #import time library for sleep function
import datetime                         #import datetime library for timestamp
import RPi.GPIO as GPIO                 #import GPIO library
GPIO.setmode(GPIO.BCM)                  #set the pins according to BCM scheme
GPIO.setup(4,GPIO.OUT)                  #configure BCM Pin #4 as OUTPUT
GPIO.setup(17,GPIO.IN)                  #configure BCM Pin #17 as INPUT
i=0; n=5; delay=20                      #limit number of tries to 5 (initially set it to 1 for debugging)
while i<n:
        LED1=GPIO.input(4)                      #read what BCM Pin #4 is set to (LED1)
        SW1=GPIO.input(17)                      #read the status of BCM Pin #17 (SW1)
        data = {'username': 'ben', 'password': 'benpass', 'SW1': SW1, 'LED1': LED1}
        res = requests.post("https://yourdomain.000webhostapp.com/scripts/sync_rpi_data.php", json=data)
#in case of errors (especially, syntax) , you may want to print res.text and comment out the statements below
        r = res.json()
        ts = datetime.datetime.now()        #get the time stamp
        print "==============Server Response at " + str(ts) + "=============="
        if r['success']==1:
        print "+++++Server request successful: "
                if LED1!=r['LED1']:
                        print "Changing LED status as requested by the server"
                        if r['LED1']==1:
                                GPIO.output(4,GPIO.HIGH)
                        else: GPIO.output(4,GPIO.LOW)
                        print "The status of LED1 is " + str(r['LED1'])
                        print "The status of SW1 is " + str(r['SW1'])
        else: print ">>>>> Server request failed - Error #" + str(r['error'])
        time.sleep(delay)                   #wait for delay seconds before sending another request
        i+=1
GPIO.cleanup()
```

Write the php script given in the next section on the server side *and then return* to do the following for testing interaction between RPi and the server:

You may want to run ***p2_t8.py*** in the background (so that you don't lose control of the terminal) and verify that it correctly sets the status of device ***SW1*** in DBOR as you change SW1 on the breadboard during the intervals (***delay*** second):

#python p2_t8.py &

Test the interaction as follows:

a) Change the SW1 on the breadboard (from on to off or off to on), and check if DBOR correctly displays the changed status.

b) Change the status of LED1 in DBOR (from on to off or off to on), and check if the LED on the breadboard powers on or off according to the value dictated by DBOR.

7) PHP script to interact with RPi: The php script, ***sync_rpi_data.php*** in ***scripts*** folder on the server, shown below responds to ***JSONRequest*** from RPi. Specifically, the following interaction happens:

> ***JSONRequest****, received by the server, is initiated by RPi and carries the status of LED1 and SW1. RPi expects the server to set the status of SW1 in table* ***device*** *as it appears in the request. In response, the server makes the changes in DBOR, and sends back the status of LED1 and SW1 as they appear in table* ***device*** *(after making the changes).*

Install the following script as ***sync_rpi_data.php*** in folder ***scripts***:

```
<?php
require_once __DIR__ . '/../../required/db_connect.php';
$input = file_get_contents("php://input");
$error=0; $out_json = array(); $out_json['success'] = 1; //assume success
$SW1_status=0; $LED1_status=0;
if ($input) {
    $json = json_decode($input, true);       //check if it json input
    if (json_last_error() == JSON_ERROR_NONE) {
        if (isset($json["username"]) && isset($json["password"]) && isset($json["SW1"])
               && isset($json["LED1"])) {
            $in_username = $json["username"];
            $in_password = $json["password"];  //if the expected fields are not null, get them
            $in_SW1 = $json["SW1"];
            $in_LED1 = $json["LED1"];
            if ($stmt=$mysqli->prepare("SELECT password FROM webuser WHERE pname = ? LIMIT 1")) {
                $stmt->bind_param('s', $in_username);
                $stmt->execute();  $stmt->store_result();   //store_result to get num_rows etc.
                $stmt->bind_result($db_password);    //get the hashed password
                $stmt->fetch();
                if ($stmt->num_rows == 1) {          //if user exists, verify the password
                    if (password_verify($in_password, $db_password)) {
                        $stmt->close();
                        if ($stmt = $mysqli->prepare("UPDATE device set status=?
                                                      where devname = 'SW1'")) { //update SW1
                          $stmt->bind_param('i', $in_SW1); $stmt->execute();
                        } else {$error=1;}
                        $stmt->close();
                        if (!$error && ($stmt = $mysqli->prepare("SELECT status FROM device
                                                      where devname = 'SW1'"))) { //read SW1
                          $stmt->execute(); $stmt->bind_result($SW1_status); $stmt->fetch();
                        } else {$error=2;}
                        $stmt->close();
                        if (!$error && ($stmt = $mysqli->prepare("SELECT status FROM device
                                                      where devname = 'LED1'"))) { //read LED1
                          $stmt->execute(); $stmt->bind_result($LED1_status); $stmt->fetch();
                        } else {$error=3;}
                        $stmt->close();
                    } else {$error=4;}
                } else {$error=5;}
            } else {$error=6;}
        } else {$error=7;}
    } else {$error=8;}
} else {$error=9;}
if ($error){
   $out_json['success'] = 0;  //flag failure
}
$out_json['SW1'] = $SW1_status; $out_json['LED1'] = $LED1_status;
$out_json['error'] = $error;  //provide error (if any) number for debugging
echo json_encode($out_json);  //encode the data in json format
?>
```