**A close up of a logo

Description automatically generated**

**CS 890EP - Internet of Things**

**Supervised by**

**Dr. Maher Elshakankiri**

**A hand holding a cellphone

Description automatically generated**[4]

**Smart Lunch Box**

**Submitted by**

**Keerthi Mettu**

**200416252**

**SMJ102@uregina.ca**

**Introduction:**

Over the past few decades technology has been growing rapidly. Internet of things is a necessary tool for today world which we use in our present life and a lot of applications improved efficiency by using it. IoT (Internet of Things) is utilized in multiple technologies to solve various problems which exist in today’s world. The unique part of IoT applications is that all the devices are connected over the internet, controlled and monitored remotely.

To illustrate, various technologies involved in enabling the Internet of Things are Radio Frequency Identification (RFID), Near field communication (NFC), Wireless Sensor Network (WSN), Different Networks for communication, Augmented Intelligence, Augmented Behavior.

Of all these technologies, WSN plays a prominent role in the development and growth of IoT applications. A wireless sensor network is defined as a group of Wireless devices that are connected over the network [5]. These devices can sense and monitor the real-time environment as well as exchange information between nodes. Each node has a predefined task. WSN consists of a base station, nodes which are sensors. Sensors sense the data and forward the data to the base stations. Overall tasks performed by wireless sensor networks is to gather the sensed data, Process the data, transfer the data using packets. Depending upon the application appropriate sensors like Temperature Sensor, Light Sensor, Pressure Sensor, Humidity Sensor, Touch Sensor, Tilt Sensor, IR Sensor, etc. are used.

Healthy human survival and lifestyle depend on the food we eat. Mindful eating is very much required for anyone who wants to take control of the food intake. Calories are the basic measuring unit of energy produced from food and it varies for different food sources. Mindful eating not only involves the calorie check on the food, but also the measure of nutrition values absorbed from them. Basically, the food components are classified as carbohydrates, proteins, and fats. Keeping a count on the amount of these components accurately on a routine basis helps the individuals to monitor the calorie intake. This is also a major requirement for any person who wants to follow a diet and maintain it as a conscious eating habit.

There are several applications on the internet to identify the calorie count for the food that is chosen. Although there are smart devices that aid in the temperature control of the food, there are certain limitations in terms of availability, accuracy and safety. Some devices like smart lunch boxes have to be introduced that can address the technical requirements of the users using necessary. A smart lunch box can be created as an Internet of Things device, that is capable of notifying the user about their lunch timings through an alarm, calculating the calories of the food present in the box and heating the food. For the sake of kid’s nutrition, it should be designed in a way to identify the food completion level in the lunch box along with the authentication, so that the box opens only for the respective users.

**Keywords:** Weighing module, temperature sensor, WIFI module, touch sensor, Alarm, Buzzer.

**Related work:**

Albert tian proposed a design for a lunch box, where it has a heating facility to warm up the food before eating [1]. This smart box uses a positive temperature coefficient heater as a heating element, which has an open loop control and doesn’t need an external diagnostic. Prepd describes a Prepd app [2], that gives information about the food, that can be prepared for a week before it starts. It helps to give the measures of the food nutrients that will be taken by the user and a broad range of diets, appetites, and health goals. Silbo Tech [3] designed a lunch box with the features like heating and cooling techniques, Bluetooth connection for phones to get notifications regarding the lunch box state and the charging feature like a power bank to electronic devices through USB connections. It also includes a safety sensor and a temperature sensor to get to know the temperature of food inside the lunch box.

**IMPLEMENTATION:**

**Problem Statement:**

Intake of calories and food consumption affects a person’s healthy lifestyle and living. Proper calorie intake and nutrition play an important role in one's diet. Now a days, being in a busy lifestyle, most of the people are unable to concentrate on the food they are consuming. Even the ones who want to become health conscious, are trying to carry their own food prepared, boxes, but they are not able to get the exact amount of nutrition acquired from it. Also, due to hustle life, people will never take note of the number of calories they take on per meal.

Most of the software applications that do the calorie sum of the food, ask for the quantity of the respective food item. If we don’t provide these values accurately, there would be variations in the amount of calories consumption for that day. It would not help a person to keep track of their diet in a systematic manner. It is also a major drawback for the ones who have goals without the help a nutritionist on weight loss or weight gain nutrition plan.

A lot of students bring their own lunch boxes to school and most employees prefer carrying their home food. Sometimes there won’t be any microwave or outlets to warm up their food. It also has a risk of food stewing in its own moisture, which results in the food becoming soggy and unappetizing.

On the other side, parents who send lunch boxes to their kids are very much concerned about their nutrition. It is always a big quest for them to know whether their kid actually had lunch or just threw the food to the dustbin.

It is really important to maintain a scheduled food time every day, in order to have an undisturbed biological clock in the body. Usually, people will forget to have their lunch on time due to the heavy workload. They need regular reminders about their lunch timings. A smart lunch box can be designed to solve all the above-mentioned problems.

**Methodology:**

A smart lunch box is designed to address all the issues described in the problem statement. As the IoT is an emerging technology with various sensors, a smart lunch box is designed using this technology. The term “smart” itself indicates that the lunch box is designed to act intelligently to maintain a conscious eating habit for a respective group of users like adults and kids. Following are the different modules implemented as a part of a smart lunch box.

1. *Weight the food in sections:* A smart lunch box is designed in such a way that many different size compartments are present inside it. Every compartment can go with a single food item. Each compartment has its own weighing module, the primary component of this module is load cell sensor along with a force sensor so accurate weight of each component can be obtained. These values are populated to the app where they are used in the next module.
2. *Calorie calculator:* The weight noted in the initial module is supplied as the input for this module. In this module, all the information about each type of food along with its carbs, protein and fat percent information is stored in the database. In the designed application there will be an interactable picture of smart box compartment. When an item is placed in the smart lunch box compartment, its respective weight will automatically be reflected in the designed app. Through this, the user has to enter the food item name manually by selecting the compartment. If the food item is already loaded in the database, the app checks its respective calories and gives accurate information about the carbohydrates, proteins, fats and other nutritional information in the form of a percentage. A final sum of calories per meal can also be indicated at the end of entering the food data.
3. *Automatic Heating Control:* Smart lunch box should have a heating element where it warms up the food present in lunch boxes by pressing the button. Resistors coil can be made and surrounded around the compartments present in the lunch box. A particular time is set through an app for heating up their smart box to warm up the food in the container. This solves the issue for not waiting too long for the microwave to heat their lunch boxes.
4. *Alarm signaling:* This module is designed especially for the people who are busy in their life as they forget about their intake of food. An alarm will help the user to remember to take their food at lunchtime. Buzzing can be stopped by using a button present on the lunch box or by operating it through the app. This alarm can also be used for different purposes like buzzing to clean the container and trigger to take the lunch box before leaving the house.
5. *Touch Authentication:* This module is specially designed for school going kids. As kids don’t have control on losing their boxes, there may be cases that their friends may take away their lunch boxes without any indication. Apart from this as this is a smart lunch box, the box should be opened in a smart manner rather than using the traditional box open method.
6. *Empty Box Notification:* This is a unique module for parents who are more concerned about their children. This would let them know as a notification in the app whether their kid is having the meal or throwing away the food from the lunch box.

**DETAILED DESCRIPTION – Module 2:**

This section of detailed description focuses on the building of the Calorie calculator (2nd module) explained in the methodology section. In order to build this model, a website for software and Arduino microcontroller’s hardware equipment is required.

In this module, the weight that is captured in the first module (Weigh the food in sections) through the load sensor is taken as an input. When the Wi-Fi module (NodeMCU) is on, this would let the transfer of weight information to the web server. Here this web server is created in a local personal computer and it would help in hosting the web application. This application is internally integrated with the database platform provided by the server environment. The DB platform will hold a specific database called Smart Lunch Box with the most popular food that is being served inside the lunch box according to the user interests. The client here could be a mobile or a computer that has an internet connection which is in the same network as that of the web server. The devices that are connecting to the web server has to be in a single network because the web server application is hosted in a local machine. As the NodeMCU sends the real time information of the load sensor, it is good to reach out the web server from any client before the start of eating process. This ensures that real time data is captured at the right moment to get the accurate calorie values that were consumed. Along with the calories, a clear view of the percentage values of carbohydrates, proteins and fat values are displayed in a tabular format.

**Hardware Components:**

Arduino Uno board

NodeMCU (Wi-Fi module)

USB cable

**Software Requirements:**

Arduino IDE

PHP editor

Apache server

MyPhpAdmin DB

**Block Diagram:**

Post sensor data via http

Web Server

NodeMCU

http request http response

Load Module attached to Arduino

Web Client

SMART LUNCH BOX

**Construction Steps:**

The interaction between a web server and a web client is made is briefly described in the following flow chart. The work flow between client and server with validations are included in between to avoid errors when communicating with the database through using the inputs provided by the user.

A close up of text on a white background

Description automatically generated

A close up of text on a white background

Description automatically generated

**Flow chart for client and web server communication**

This module solely deals with the design and development of website application from scratch and hosting it in a local web server environment. NodeMCU is programmed in Arduino IDE to connect to the Wi-Fi using a http request to the server. The IP address of the web server is displayed to the user so that, pinging the address would give the user reach the web application. When this server is up and running, this will host the created web application. The application is created to choose the food items that are placed in the lunch box container. It enables the users to choose the items that are pre-determined in the website and helps to calculate the calories of the food. The website application takes care of the handling the input values from the previous module, capturing the user given food variables and retrieve respective standard values for calculations in the back end. All the validations are taken into consideration for the security of the data values provided by the user in the form applet present in the first page of the application.

**Hardware Setup:**

NodeMCU is connected to the laptop using the USB cable so that it has Arduino IDE sketch program running on it. This program in the IDE is responsible for connecting with the Wi-Fi network of the user. Once the NodeMCU gets connected with the network, the Arduino micro controller attached to it, will retrieve the weight of the food that is calculated from the module 1 of this project (Weigh the food in sections). These weight values are published to the web application when the NodeMCU connects with the Wi-Fi network and opens the web application developed with the php scripting language.

The NodeMCU is further connected to Arduino board for serial connection of the data exchange between the Arduino software and the Arduino board. To achieve this, the data pins on NodeMCU like D5 and D6 are connected to the digital pins 5 and 6 on the Arduino board.

A close up of electronics

Description automatically generated[6]

**Environment and Database setup:**

The PC chosen for web server setup should be installed with the Apache server environment to host the web applications and MyPhpAdmin software files to run the database along with the server. The web application’s front end is developed with the html scripting language and back end communication with the php scripting language. MyPhpAdmin is a database server that uses MySQL querying operation. Initial setup involves providing configurations and authentication user credentials which are further used in the connection process of the server host with the created databases.

A database named “SmartLunchBox” is created with the table called Food. It has been filled with column values like food item names with respective calories, carbs, protein and fat percentage values according to a given standard weight.

**Design challenges:**

* Initially ESP8266 Wi-Fi module was ordered and tried multiple sample examples to show the connection with Wi-Fi. In this process, it is understood that there are 6 different types of ESP8266 modules, and it is one of the components that has specific library functions (Phpoc.h, SPI.h, SoftwareSerial.h and ESPWiFi.h) that communicates with the connection and it will work only for their specific modules. Also, it has a restriction of 3.3V power supply use where it does not work when this is connected directly with the 3.3V digital pin on Arduino board. It would need an external voltage adapter that can convert a 5V to 3.3V. These are some unresolved problems left in the Arduino forums and other web forums [8].
* With the NodeMCU, the sample examples were getting compiled, but the port connecting the NodeMCU with the laptop was not detecting. This problem is encountered only when it is in connection to MacBook Air PC unique port driver configurations. It took a long while to understand this until it was tested on a Windows PC.
* Incompatible external server application installations on the MacBook are always a failure. XAMPP server is a combined environment for web application hosting and database (My SQL) integration. But the series of steps followed on its installation process are not exactly matching with the flow steps that are restricted by the laptop. The steps mentioned are not exactly replicable and navigates to the different pages on error creation [9] [10].
* On finding the alternate solutions for server environment setup in MacBook Air, it is identified that there is an in-built php server platform provided by the Mac OS and it has to be configured from its configuration files (.conf) through shell commanding on the Mac terminal. It took a long trail and error process in modifying the OS files, saving them safely and exit back to the terminal without errors. This is because some of the steps are clearly not explained anywhere except for the latest version of the Mac OS. Author in medium.com (Jangid, 2018) was a great help to achieve these OS changes to a greater extent. [7]
* There was a bigger difficulty in connecting the NodeMCU board with the Arduino UNO board. To establish serial communication between both of them, there was no major solution providers online except the author from mybtechprojects.com [12]. It is a main advantage to post the sensor information or the control break passage from one board to another serially.

**Test Results:**

A successful test suite works in this following sequence:

1. NodeMCU connects the Wi-Fi network provided by the user.
2. User will be able to open the web application through the IP address ping.
3. User can view the measured weight values on the web page.
4. On providing the food item names, click on the submit button.
5. Validations are performed on the user inputs.
6. On passing these validations, user will be navigated to another page to view the nutrition facts of the given food in a tabular format.

**DETAILED DESCRIPTION – Module 4:**

This section of detailed description focuses on the building of the Alarm Signaling (4th module) explained in the methodology section. In order to build this model, an alarm and its control has to be included in the hardware system.

Timely intake of food is a crucial aspect to any person who follows diet control. Mindful eating with the help of calorie calculator and time to time alarm signal for consuming the lunch helps to users to have a conscious eating. In this module, the alarm will ring only there is a pre-determined time that is system. Whenever the system identifies that this time is met, the alarm would start buzzing to remind its user about the lunch timing. To stop this alarm, there is a button provision included on the lunch box and user turn it off. If the user is away from the box and to turn it off immediately, he/she can open the localhost webpage either from the desktop and mobile to click on a link which also does the alarm ring stopping. To use this virtual control, the lunch box and the user has to be connected to the same Wi-Fi. And the IP of the lunch box has to pre-identified and coded in a way that user is able to use the alarm switch control from the localhost web page. An additional LED identification on the lunch box can be controlled using the same web operation methodology and this can be used as an aid for people with audibility problems. This alarm can be used not only for notifying the lunch time to the user but also to clean or do any maintenance activity for the lunch box.

**Hardware Components:**

Arduino Uno board

NodeMCU

Buzzer

Push Button

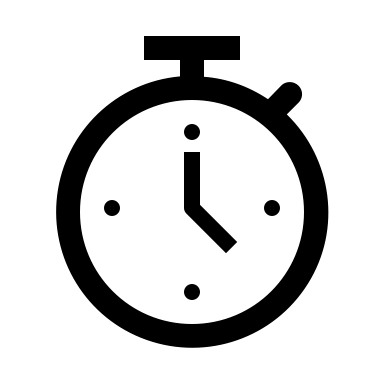
Resistor 200ohms

Jumper wires

Bread Board

USB cable

**Block Diagram:**



Client Localhost device

Button

Controls on/off

Smart Lunch Box

**Design of Alarm Signaling:**

* The hardware is included inside the lunch box.
* Set a pre-defined time for the lunch box so that it should remind the user about the lunch timing.
* The buzzer rings according to the programmed set time interval.
* The push button arranged on the lunch box can be pressed to turn off the ringing buzzer.
* Multiple alarms can be set on this lunch box, so that user can be intimated about any reminder that they would like to know.

A circuit board

Description automatically generated

**Construction steps:**

* The alarm buzzer and the push button are directly connected with the NodeMCU, which is further serially connected with the Arduino UNO.
* The positive leg of the buzzer is connected to the data pin D0 in the NodeMCU and negative leg of it to the central GND connection used on the breadboard.
* The one end of push button is used for positive connections and another end with the GND signaling. One of the positive end pins is connected with the D1 data pin on the NodeMCU and another positive pin with the power supply that provides 5V source to the push button. The negative end is given a 200ohms resistance through a resistor before giving the ground rail connection.
* The NodeMCU and Arduino UNO are serial connected using two more jumper wires that are attaching the data pins D5 and D6 on the NodeMCU with the digital pins 5 and 6 respectively on the Arduino UNO board.

**Test Results:**

A successful test suite works in this following sequence:

1. A pre-determined first alarm is set in the programming to start buzzing after 10 minutes on turning on the lunch box connection with the Wi-Fi and localhost.
2. A second alarm is set to ring after completing 30 minutes on the start of program running.
3. On completing 10 minutes, the first alarm began to ring.
4. Clicking on the push button, the alarm stopped ringing.
5. The second alarm also starting buzzing successfully after completing of 30 minutes time.
6. Hitting the link on the hyperlink in the localhost page, turned off both the alarm buzzing and the LED light off on the NodeMCU.

**Screenshots:**

**1.**

A screenshot of a cell phone

Description automatically generated

NodeMCU on successful connection with the user Wi-Fi network, is displaying the IP value in the serial monitor. This program code is from reference [11].

2.

A screenshot of a cell phone

Description automatically generated

SmartLunchBox Database and Food table data

3.

A screenshot of a cell phone

Description automatically generated

Display of weights on opening the web application (Initial development phase)

4.

A close up of a logo

Description automatically generated

Submitting the items to view the results in the following page (final development)

5.

A screenshot of a cell phone

Description automatically generated

Second page in web application showing the food nutrients details

6.

A screenshot of a cell phone

Description automatically generated

The serial monitor output when localhost is successfully opened and operated by the users present in the same network.

7.

A screenshot of a cell phone

Description automatically generated

The final view of the local host with alarm turn off link connection

8.A screenshot of a cell phone

Description automatically generated

When the alarm starts to ring, the output view on the serial monitor and when stopped.

9.

A close up of a computer

Description automatically generated

The complete hardware connection used for construction of both the modules.

**Future work:**

On a small scale, all the individually developed modules can be integrated with each other and can come up with the complete set up of the hardware in the smart lunch box. Alongside these modules, other features like, power bank supply from the smart lunch box can be provided for the other portable devices through USB connections. To implement this module, self-charging system has to be included within the model. An LCD display on top of the lunch box can make the system more interactive with the end user and can show appropriate messages of the functions that are happening currently or the background actions that needs user attention. An auto cleaning system with in the smart lunch box can also be implemented which can reduce the manual efforts and adds more smartness for the lunch box. The last two features could need a largely scalable and efficient hardware for actual implementation as the sensor information should be intelligent enough to take decision-based actions even before performing.

**Conclusion:**

The final report of this project presents the individual development of two different modules (Calorie calculator and Alarm signaling). The detailed implementation of both the described modules helps for the combination of the rest modules in the smart Lunch Box. The first module in this Smart Lunch Box, lets the user be conscious on their food intake by automatically measuring the food weight and doing the calories calculating for the food present in the compartmentalized smart box. So, the implementation of the first module is integrated with the second module, calorie calculator which helps the user to see their food weight both virtually and live when devices being present in the same network. The Automatic heating and Alarm signaling helps the users to enjoy the warm food on the right time. The alarm signal here can be used for multiple purposes based on the user preferences to remind about anything and can have control of switching it off through hardware interaction or virtually from the web page interaction by the user. Touch authentication and empty box notification would be more useful for the parents having small kids to keep a track on their food intake. All the modules, that are implemented has so far produced the positive results, when all the negatives are taken care off in the program codes. The main aim of the Smart lunch Box project is achieved individually by the project developers. As there is a drawback on minor integrations with the other modules, the Arduino is fluctuating the power supply for the hardware components and also creating memory maximum utilization problems in the programming Arduino IDE. So, it’s the only reason for not being integration the modules with one another. On a whole, this Smart lunch Box assists its users to manage a balanced diet and keeps track of the food consumption with all its customizable smart features with respect to the user’s need.

**Appendix:**

**Code:**

Arduino Integration.ino

#include <SoftwareSerial.h>

#include <ArduinoJson.h>

#include "HX711.h"

#define DOUT A0

#define CLOCK A1

#define DOUT2 A2 // Load Cell B pin connections

#define CLK2 A3

SoftwareSerial s(5, 6);

HX711 scale;

HX711 scale1;

float calibration\_factor = -160.20; //-106600 worked for my 40Kg max scale setup

float calibration\_factor1 = -58.11;

void setup() {

s.begin(9600);

scale.begin(DOUT, CLOCK);

scale1.begin(DOUT2, CLK2);

scale.set\_scale(-160.20); //Calibration Factor obtained from first sketch

scale1.set\_scale(-58.11);

scale.tare(); //Reset the scale to 0

scale1.tare();

}

StaticJsonBuffer<1000> jsonBuffer;

JsonObject& root = jsonBuffer.createObject();

void loop() {

int weight1 = scale.get\_units();

int weight2 = scale1.get\_units();

delay(2000);

root["Foodweight1"] = weight1;

root["Foodweight2"] = weight2;

Serial.println(Foodweight1);

Serial.println(Foodweight2);

if (s.available() > 0)

{

root.printTo(s);

}

}

**NodeMCU\_Integration.ino**

#include <ESP8266WiFi.h>

#ifndef STASSID

#define STASSID "Access4335"

#define STAPSK "6399974545"

#endif

const char\* ssid = STASSID;

const char\* password = STAPSK;

const int buzzer = 5;

int temp = 0;

long interval = 1 \* 30 \* 1000; //1minute

unsigned long previousMillis = 0;

int ring = 0;

const int button = 16;

// Create an instance of the server

// specify the port to listen on as an argument

WiFiServer server(80);

void setup() {

Serial.begin(115200);

pinMode(buzzer, OUTPUT);

pinMode(button, INPUT);

// prepare LED

pinMode(LED\_BUILTIN, OUTPUT);

digitalWrite(LED\_BUILTIN, 0);

// Connect to WiFi network

Serial.println();

Serial.println();

Serial.print(F("Connecting to "));

Serial.println(ssid);

WiFi.mode(WIFI\_STA);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(F("."));

}

Serial.println();

Serial.println(F("WiFi connected"));

// Start the server

server.begin();

Serial.println(F("Server started"));

// Print the IP address

Serial.println(WiFi.localIP());

}

void loop() {

temp = digitalRead(button);

unsigned long currentMillis = millis();

int Foodweight1 = 120;

int Foodweight2 = 140;

if (currentMillis - previousMillis > interval)

{

if (ring == 0) {

digitalWrite(buzzer, HIGH);

delay(500);

Serial.println("RINGING...");

// digitalWrite(buzzer, LOW);

// delay(500);

if (temp == HIGH)

{

//Serial.println("Turning off the buzzer");

ring = 1;

digitalWrite(buzzer, LOW);

Serial.println("stopping the buzzer");

}

}

}

if (temp == HIGH)

{

//Serial.println("Turning off the buzzer");

ring = 1;

digitalWrite(buzzer, LOW);

//Serial.println("changed variable to 1 in block 2");

}

// Check if a client has connected

WiFiClient client = server.available();

if (!client) {

return;

}

Serial.println(F("new client"));

client.setTimeout(5000); // default is 1000

// Read the first line of the request

String req = client.readStringUntil('\r');

Serial.println(F("request: "));

Serial.println(req);

// Match the request

int val;

if (req.indexOf(F("/gpio/0")) != -1) {

val = 0;

} else if (req.indexOf(F("/gpio/1")) != -1) {

val = 1;

} else if (req.indexOf(F("/gpio/2")) != -1) {

ring = 1;

} else {

Serial.println(F("invalid request"));

val = digitalRead(LED\_BUILTIN);

}

// Set LED according to the request

digitalWrite(LED\_BUILTIN, val);

// read/ignore the rest of the request

// do not client.flush(): it is for output only, see below

while (client.available()) {

// byte by byte is not very efficient

client.read();

}

// Send the response to the client

// it is OK for multiple small client.print/write,

// because nagle algorithm will group them into one single packet

client.print(F("HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>\r\nGPIO is now "));

client.print((val) ? F("low") : F("high"));

client.print(F("<br><br>Click <a href='http://"));

client.print(WiFi.localIP());

client.print(F("/gpio/1'>here</a> to switch buzzer and LED off, or <a href='http://"));

client.print(WiFi.localIP());

client.print(F("/gpio/0'>here</a> to switch LED on.</html>"));

client.println(F("<br><br><br> Click for web pages <a href='http://localhost/~keerthimettu/page1.php'> link </a>"));

client.println("<br><br>");

client.print("Foodweight1 is : ");

client.println(Foodweight1);

client.print(" ");

client.print("Foodweight2 is : ");

client.println(Foodweight2);

// The client will actually be \*flushed\* then disconnected

// when the function returns and 'client' object is destroyed (out-of-scope)

// flush = ensure written data are received by the other side

Serial.println(F("Disconnecting from client"));

currentMillis = previousMillis;

}

Page1.php

<?php

$text = explode('Foodweight1 is : ', $homepage, 2)[1];

//echo $text. "<br>";

//print\_r($text);

$x=explode(" ",$text);

//print\_r($x);

$fw1=$x[0];

$fw2=$x[4];

?>

<!DOCTYPE html>

<html>

<head>

<title> Nutrition facts </title>

<style>

body {

background-color: black;

background-image: radial-gradient(lightblue, black);

text-align: left;

color: white;

font-family: Arial, Helvetica, sans-serif;

}

.button {

background-color: #f4511e;

border: none;

color: white;

padding: 16px 32px;

text-align: center;

font-size: 16px;

margin: 4px 2px;

opacity: 0.6;

transition: 0.3s;

display: inline-block;

text-decoration: none;

cursor: pointer;

}

.button:hover {opacity: 1}

</style>

</head>

<body>

<strong>

<h1 align="center">Want to know your Nutrition facts?</h1> </strong>

<br>

<form method="post" id="form"> <!-- action="<?php $\_PHP\_SELF ?>" "<?php echo htmlspecialchars($text); ?>" -->

1st Weight: <input type="text" name="weight1" value="<?=$fw1?>"><br>

2nd Weight: <input type="text" name="weight2" value="<?=$fw2?>"><br>

<p>Select the item in compartment 1: </p>

<select name="com1">

<option value=""></option>

<option value="rice">Rice</option>

<option value="dal">Dal</option>

<option value="scrambled egg">Scrambled egg</option>

<option value="green salad">Veg salad</option>

<option value="pasta">Pasta</option>

</select> <?php echo $fw1; ?>

<p>Select the item in compartment 2: </p>

<select name="com2">

<option value=""></option>

<option value="rice">Rice</option>

<option value="dal">Dal</option>

<option value="scrambled egg">Scrambled egg</option>

<option value="green salad">Veg salad</option>

<option value="pasta">Pasta</option>

</select> <?php echo $fw2; ?>

<br>

<br>

<br>

<input type="submit" class="button" value="Show Results" formaction="/~keerthimettu/ss.php">

</form>

<!--onClick="document.location.href='http://localhost/~keerthimettu/ss.php'">

-->

</body>

</html>

**SecondPage.php**

<?php

$servername = "localhost";

$username = "root";

$password = "Arduino@11";

if ( isset( $\_POST['com1'])? $\_POST['com1'] : false ) {

//echo '<h2>form data retrieved by using $\_POST variable<h2/>';

}

$Item1 = $\_REQUEST['com1'];

$Item2 = $\_REQUEST['com2'];

$w1=$\_REQUEST['weight1'];

$w2=$\_REQUEST['weight2'];

echo "Items chosen are : ".$Item1.", ".$Item2."<br> Weights given are : ".$w1.", ".$w2 ;

try {

$conn = new PDO("mysql:host=$servername;dbname=SmartLunchBox", $username, $password);

// set the PDO error mode to exception

$conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

// echo " <br> Connected to db successfully"."<br>";

$riceCal = "SELECT Items,Calories FROM Food";

$result = $conn->query($riceCal);

$Item1query = "SELECT Calories,carbs,protein,fat FROM Food where Items='".$Item1."'";

$Item1Cal=$conn->query($Item1query);

$Item2query = "SELECT Calories,carbs,protein,fat FROM Food where Items='".$Item2."'";

$Item2Cal=$conn->query($Item2query);

foreach ($Item1Cal as $row)

{

//echo $row['Calories']."<br>";

$cal1=$row['Calories'];

$carb1=$row['carbs'];

$prot1=$row['protein'];

$fat1=$row['fat'];

}

// echo $Item1." calories are = ".$cal1.",carbs = ".$carb1.",protein = ".$prot1.", fat = ".$fat1;

// echo "<br> ";

foreach ($Item2Cal as $row)

{

//echo $row['Calories']."<br>";

$cal2=$row['Calories'];

$carb2=$row['carbs'];

$prot2=$row['protein'];

$fat2=$row['fat'];

}

//100gms is chosen as standard weight

$actualCal1= ($cal1\*$w1)/100;

$actualCal2= ($cal2\*$w2)/100;

$avgCarb=($carb1+$carb2)/2;

$avgProt=($prot1+$prot2)/2;

$avgfat=($fat1+$fat2)/2;

$totalCal=$actualCal1+$actualCal2;

// //fetch tha data from the database

// foreach ($result as $row) {

// echo $row['Items']." - ".$row['Calories']."<br>"; // echo "id: " .$row["Calories"]."<br>";

// }

}

catch(PDOException $e)

{

echo "Connection failed: " . $e->getMessage();

}

?>

<html>

<head>

<title>Nutrition today</title>

<style>

body {

/\* background-color: yellow; \*/

background-image: radial-gradient(red, yellow, green);

background-color: #cccccc;

text-align: center;

color: black;

font-family: Arial, Helvetica, sans-serif;

}

table, th, td {

border: 1px solid black;

border-collapse: collapse;

}

th, td {

padding: 15px;

text-align: left;

}

table#t01 tr:nth-child(even) {

background-color: #eee;

}

table#t01 tr:nth-child(odd) {

background-color: #fff;

}

table#t01 th {

background-color: black;

color: white;

}

</style>

</head>

<body>

<!-- this is second page-->

<h1>Results</h1>

<!--<p>This is a paragraph.</p>-->

<table id="t01" style="width:600px" align="center">

<tr>

<th>Item</th>

<th>Calories</th>

<th>Carbs %</th>

<th>Protein %</th>

<th>Fat %</th>

</tr>

<tr>

<td><?=$Item1?></td>

<td><?=$actualCal1?></td>

<td><?=$carb1?></td>

<td><?=$prot1?></td>

<td><?=$fat1?></td>

</tr>

<tr>

<td><?=$Item2?></td>

<td><?=$actualCal2?></td>

<td><?=$carb2?></td>

<td><?=$prot2?></td>

<td><?=$fat2?></td>

</tr>

</table>

<br>

<p> Total calories of the meal = <?=$totalCal?></p>

<br>

<br>

<table id="t01" style="width:300px" align="center">

<tr>

<th></th>

<th>Average %</th>

</tr>

<tr>

<td>Carbs</td>

<td><?=$avgCarb?></td>

</tr>

<tr>

<td>Protein</td>

<td><?=$avgProt?></td>

</tr>

<tr>

<td>Fats</td>

<td><?=$avgfat?></td>

</tr>

</table>

</body>

</html>

**References:**

1. Kickstarter. (2019). *Smart Lunchbox*. [online] Available at: https://www.kickstarter.com/projects/342822390/smart-lunchbox [Accessed 31 May 2019].
2. Kickstarter. (2019). *Prepd Pack - The Lunchbox Reimagined*. [online] Available at: https://www.kickstarter.com/projects/prepd/prepd-pack [Accessed 31 May 2019].
3. Indiegogo. (2019). *Silbo - The World's First Smart LunchBox*. [online] Available at: https://www.indiegogo.com/projects/silbo-the-world-s-first-smart-lunchbox#/ [Accessed 31 May 2019].
4. Engadget.com. (2019). *Engadget is now a part of Oath*. [online] Available at: https://www.engadget.com/2016/01/19/prepd-pack-lunchbox-healthkit/ [Accessed 31 May 2019].
5. Network (WSN) Architecture And Applications. [online] Available at:<https://www.elprocus.com/architecture-of-wireless-sensor-network-and-applications>[Accessed 22 May. 2019].
6. “Getting Started with NodeMCU Using ESPlorer | NodeMCU.” Electronicwings.Com, 2019, [www.electronicwings.com/nodemcu/getting-started-with-nodemcu-using-esplorer. Accessed 4 July 2019](http://www.electronicwings.com/nodemcu/getting-started-with-nodemcu-using-esplorer.%20Accessed%204%20July%202019).
7. Hitesh Jangid. “Install Apache, MySQL, PHP MacOS Mojave 10.14.” Medium, Better Programming, Dec. 2018, medium.com/better-programming/install-apache-mysql-php-macos-mojave-10-14-b6b5c00b7de. Accessed 3 July 2019.
8. “Send Data to a Website Using Arduino Uno.” Arduino.Cc, 2016, forum.arduino.cc/index.php?topic=378574.0. Accessed 4 July 2019.
9. “How to Install, Start and Test XAMPP on Mac OSX | Webucator.” Webucator, 2019, www.webucator.com/how-to/how-install-start-test-xampp-on-mac-osx.cfm. Accessed 4 July 2019.
10. Steve. “XAMPP Object Not Found on Mac.” Stack Overflow, 18 Aug. 2018, stackoverflow.com/questions/51910811/xampp-object-not-found-on-mac. Accessed 3 July 2019.
11. Jaya kumar, Magesh. “Control an LED from Webserver Using NodeMcu or Esp8266 Programming with Arduino IDE.” Blogspot.Com, 2014, internetofthinking.blogspot.com/2015/12/control-led-from-webserver-using.html. Accessed 3 July 2019.
12. https://www.facebook.com/My-BTech-Projects-591244817907142. “Serial Communication between NodeMCU and Arduino.” MYBTECHPROJECTS, 11 May 2019, mybtechprojects.tech/serial-communication-between-nodemcu-and-arduino/. Accessed 11 July 2019.