PRO-C252

WEATHER MONITORING SYSTEM - 2

What is our GOAL for this CLASS?

In this class, we were introduced to **flask server and API through which we fetched the sensor's data from Google firestore.**

What did we ACHIEVE in the class TODAY?

- We learned how to create a webpage
- We learned how to create a flask server

Which CONCEPTS/ CODING BLOCKS did we cover today?

- We learned how to set up a Flask server and create an API to call data from Database and display the values on HTML using the post method.
- We designed our Weather Monitoring webpage using HTML
- We learned about bootstrap
  - In Bootstrap we have precompiled style and setting files that are quick and easy to use for any web development styling, it makes your website responsive
  - Bootstrap follows a box model, and works in rows and columns.
  - The content should always be inside a column instead of directly being inside a row.
- Flask is the python module used to create instances of web applications

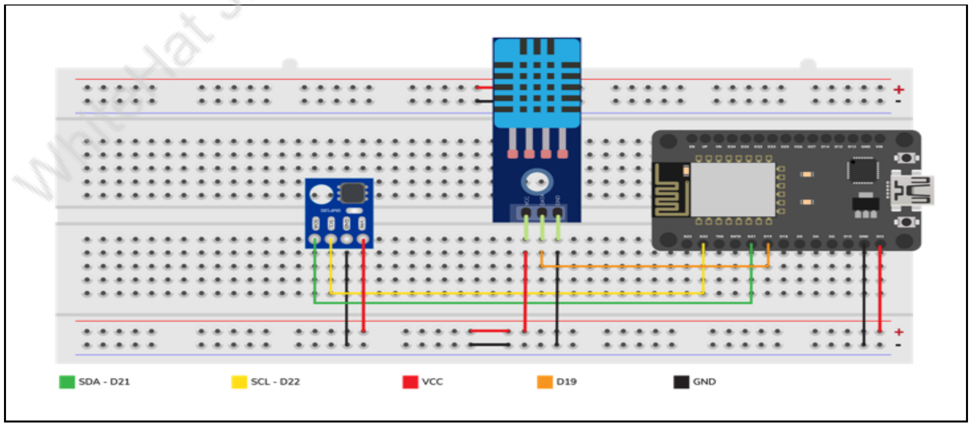
How did we DO the activities?

1. Gather the material from the IoT kit:
   - 1 x ESP32
   - 1 x USB Cable
   - 1 x Breadboard
   - 9 x Jumper wires
   - 1 x DHT11 sensor
   - 1 x BMP180 sensor

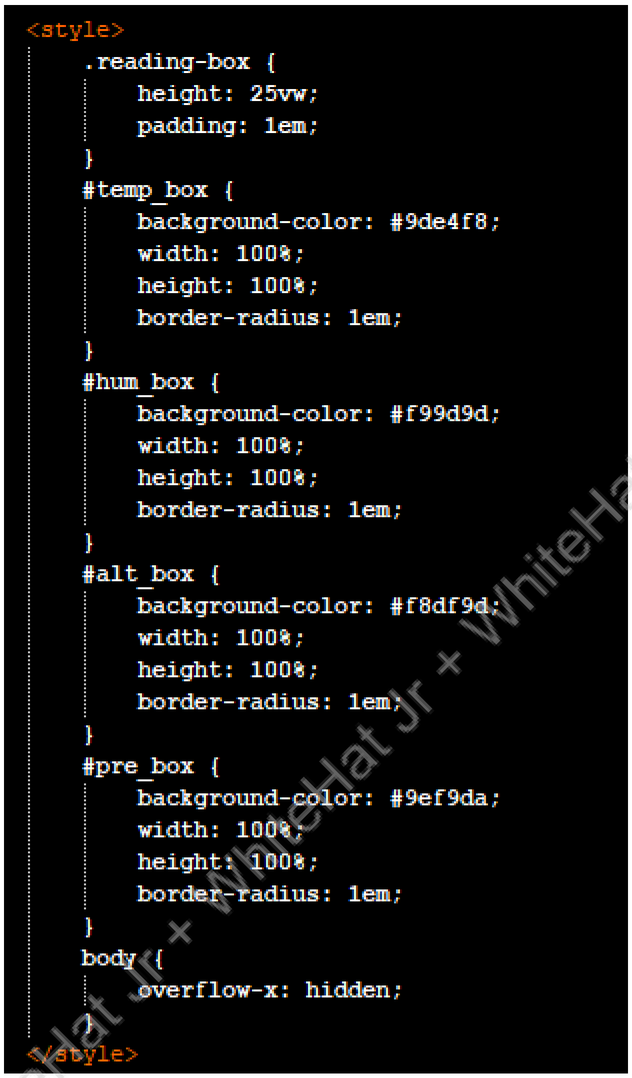
2. Connections for **Circuit** Diagram
   - **BMP180 VCC pin**: Connect with 3V3 PIN of the ESP32
   - Supply VCC(positive) from ESP32 (VIN PIN) to the breadboard positive rail.
   - Supply GND(negative) from ESP32 (GND PIN) to breadboard negative rail.

3. Connect BMP180 sensor
   - Connect VCC of BMP180 with the positive rail of the breadboard
   - Connect GND of BMP180 with the negative rail of the breadboard
   - Connect SCL pin with ESP32 pin 22
   - Connect SDA pin with ESP32 pin 21

4. **Connect DHT11 sensor**
   - Connect VCC of DHT11 with the positive rail of the breadboard
   - Connect GNDf DHT11 with negative rail of the breadboard
   - Connect Data/Outpin pin with ESP32 pin 19

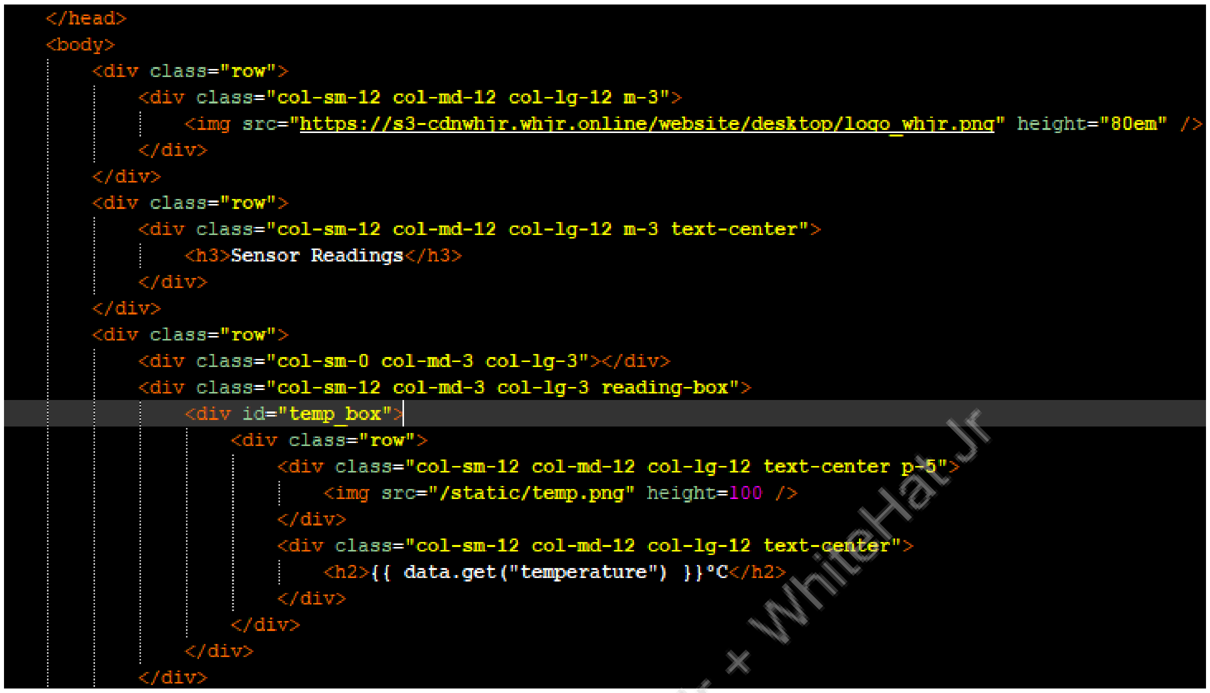


5. Design the Weather Monitoring webpage using HTML
   - Add style to the webpage in the style tag; add width, height, background color for the page.



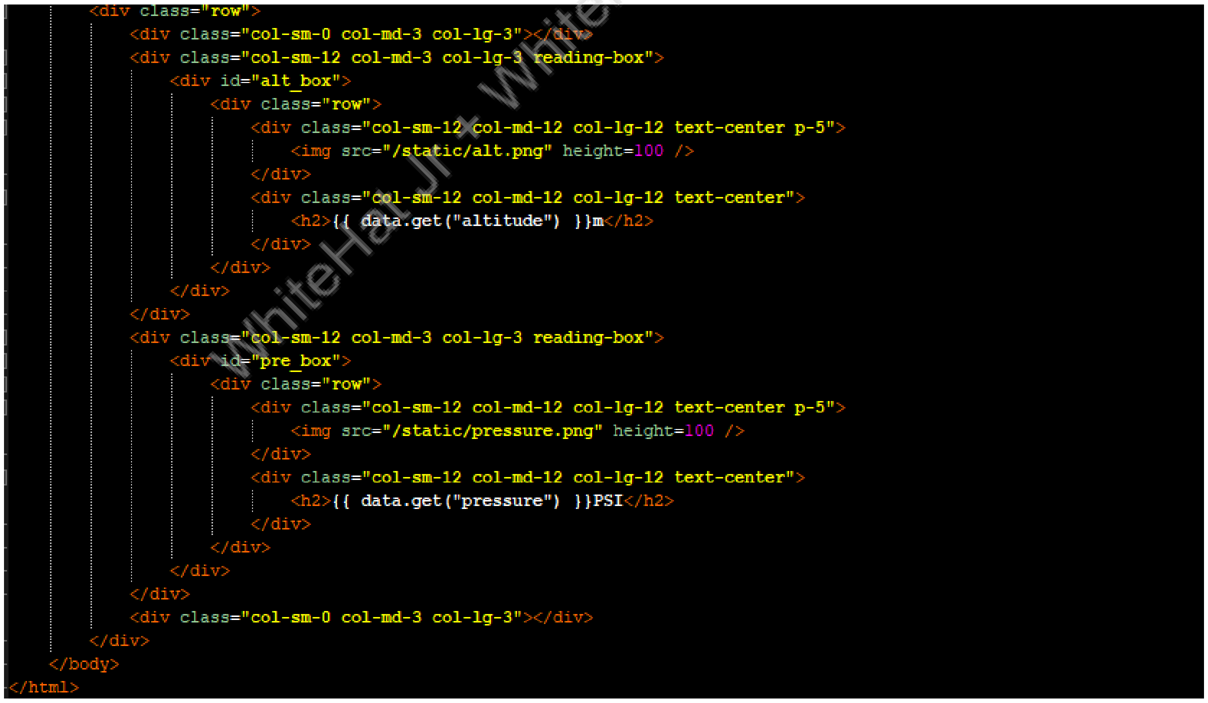
```
<style>
    .reading-box {
        height: 25vw;
        padding: 1em;
    }
    #temp_box {
        background-color: #9de4f8;
        width: 100%;
        height: 100%;
        border-radius: 1em;
    }
    #hum_box {
        background-color: #f99d9d;
        width: 100%;
        height: 100%;
        border-radius: 1em;
    }
    #alt_box {
        background-color: #f8df9d;
        width: 100%;
        height: 100%;
        border-radius: 1em;
    }
    #pre_box {
        background-color: #9ef9da;
        width: 100%;
        height: 100%;
        border-radius: 1em;
    }
    body {
        overflow-x: hidden;
    }
</style>
```

- The first **<div>** tag contains a class called **row**. This defines a bootstrap row.
- Inside this div, we have another div tag with class **col-sm-12 col-md-12 col-lg-12**
- A container can be divided into 12 different sections in terms of width.
- col defines a bootstrap column.
- sm defines column's width in small screen (mobile)
- md defines column's width in medium screen (tablet)
- lg defines column's width in large screen (desktop or laptop)
- text-center simply means to have all the text in the center of this column.
- p-3 is for padding. The number 3 here could have been anything from 1-5.
- ***col-sm-12*** have the full width of the row on the small screen, ***col-md-12*** have the full width of the row on medium screen and ***col-lg-12*** have the full width of the row on a large screen.
- Add headers for temperature, pressure, altitude and humidity.



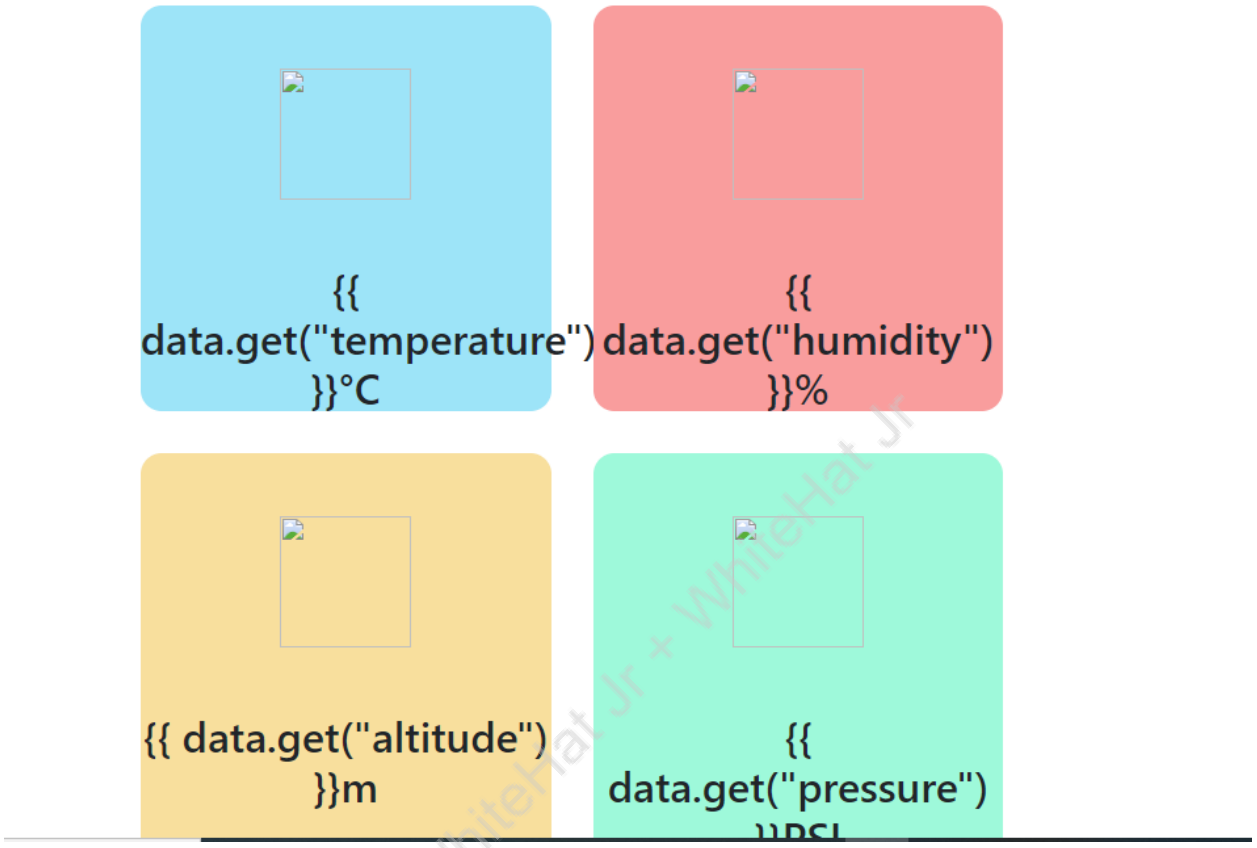
```
  </head>
  <body>
    <div class="row">
      <div class="col-sm-12 col-md-12 col-lg-12 m-3">
        <img src="https://s3-cdnwhjr.whjr.online/website/desktop/logo_whjr.png" height="80em" />
      </div>
    </div>
    <div class="row">
      <div class="col-sm-12 col-md-12 col-lg-12 m-3 text-center">
        <h3>Sensor Readings</h3>
      </div>
    </div>
    <div class="row">
      <div class="col-sm-0 col-md-3 col-lg-3"></div>
      <div class="col-sm-12 col-md-3 col-lg-3 reading-box">
        <div id="temp_box">
          <div class="row">
            <div class="col-sm-12 col-md-12 col-lg-12 text-center p-5">
              <img src="/static/temp.png" height=100 />
            </div>
            <div class="col-sm-12 col-md-12 col-lg-12 text-center">
              <h2>{{ data.get("temperature") }}°C</h2>
            </div>
          </div>
        </div>
      </div>
```

- **data.get()** is used to get value from API

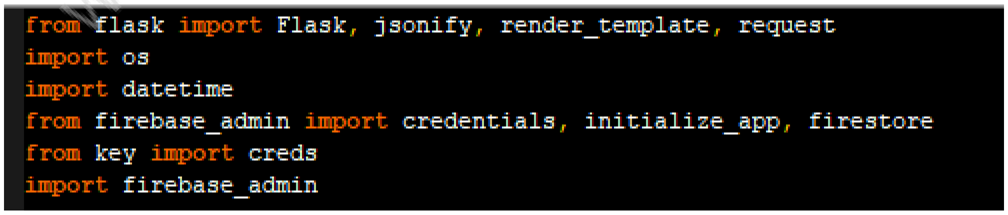


```
    <div class="row">
      <div class="col-sm-0 col-md-3 col-lg-3"></div>
      <div class="col-sm-12 col-md-3 col-lg-3 reading-box">
        <div id="alt_box">
          <div class="row">
            <div class="col-sm-12 col-md-12 col-lg-12 text-center p-5">
              <img src="/static/alt.png" height=100 />
            </div>
            <div class="col-sm-12 col-md-12 col-lg-12 text-center">
              <h2>{{ data.get("altitude") }}m</h2>
            </div>
          </div>
        </div>
      </div>
      <div class="col-sm-12 col-md-3 col-lg-3 reading-box">
        <div id="pre_box">
          <div class="row">
            <div class="col-sm-12 col-md-12 col-lg-12 text-center p-5">
              <img src="/static/pressure.png" height=100 />
            </div>
            <div class="col-sm-12 col-md-12 col-lg-12 text-center">
              <h2>{{ data.get("pressure") }}PSI</h2>
            </div>
          </div>
        </div>
      </div>
      <div class="col-sm-0 col-md-3 col-lg-3"></div>
    </div>
  </body>
</html>
```

- Read your HTML file it will look like this.

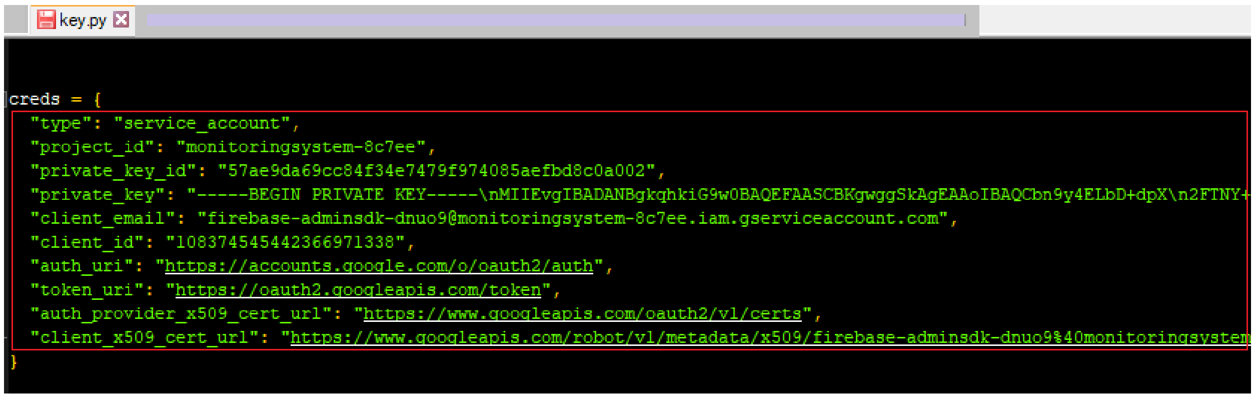

{{ data.get("temperature") }}°C
{{ data.get("humidity") }}%
{{ data.get("altitude") }}m
{{ data.get("pressure") }}PSI

6. Set up the server and fetch the real-time values through API
    - "render_template" method from the flask framework, pass an HTML file to the method and it returns to the browser when the user visits the "URL" associated with that template.
    - firebase_admin:This module contains functions and classes that facilitate interacting with the Firebase Realtime Database



```
from flask import Flask, jsonify, render_template, request
import os
import datetime
from firebase_admin import credentials, initialize_app, firestore
from key import creds
import firebase_admin
```

- **key.py** is the file where we will save creds of Google firebase



key.py

```
creds = {
  "type": "service_account",
  "project_id": "monitoringsystem-8c7ee",
  "private_key_id": "57ae9da69cc84f34e7479f974085aefbd8c0a002",
  "private_key": "-----BEGIN PRIVATE KEY-----\nMIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQCbn9y4ELbD+dpX\n2FTNY+
  "client_email": "firebase-adminsdk-dnuo9@monitoringsystem-8c7ee.iam.gserviceaccount.com",
  "client_id": "108374545442366971338",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/firebase-adminsdk-dnuo9%40monitoringsystem
}
```

- To avoid conflicts with libraries, install Flask in a virtual environment.

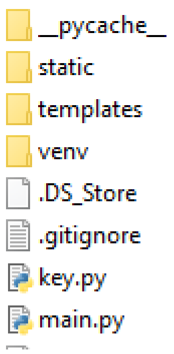
- To create and activate a virtual environment, use following commands -
    - Mac/Ubuntu -
    - python -m venv venv
    - source venv/bin/activate

    - Windows -
    - python -m venv venv
    - venv\Scripts\activate.bat

- run the command to install flask into our virtual environment
    - **pip install flask**

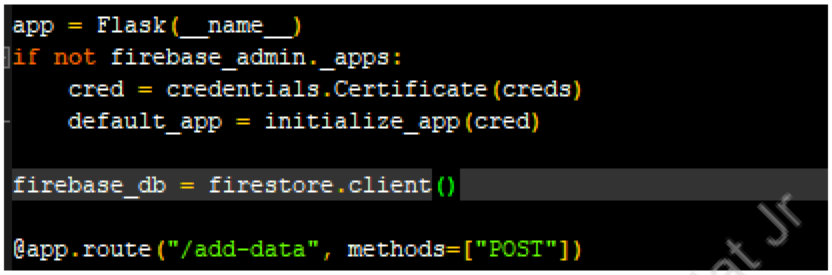
- run the following command to install **firebase_admin** into our virtual environment
    - **pip install firebase_admin**

- The flask framework looks for HTML templates in a folder called ***templates***. Create a folder called "***templates***" which will contain HTML pages. Here is how the web app directory tree should be like at this point:
- Python or .py script stays outside of the templates folder.



```
__pycache__
static
templates
venv
.DS_Store
.gitignore
key.py
main.py
```

7. To start the application, call the main function
    - Add google firestore credentials
    - Store **firestore cedentilas** in variable **cred**
    - **credentials.Certificate function** is used for authentication whth google firebase
    - Call the **add data** API ans using **POST** method display on HTML page



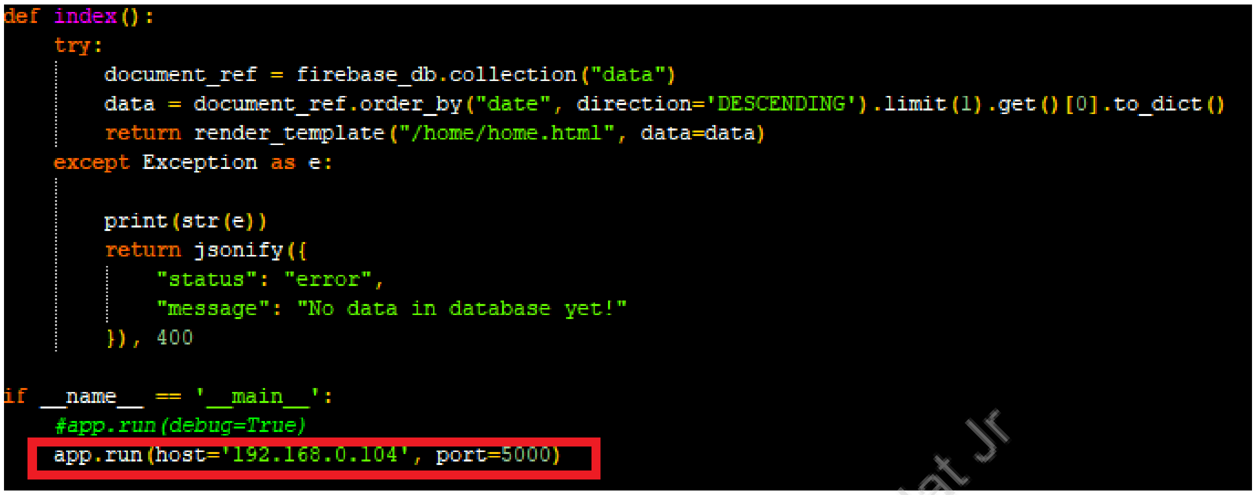
```
app = Flask(__name__)
if not firebase_admin._apps:
    cred = credentials.Certificate(creds)
    default_app = initialize_app(cred)

firebase_db = firestore.client()

@app.route("/add-data", methods=["POST"])
```

    - make function add_data()
    - Make variable to save values for **temperature, humidity, pressure and altitude.**
    - **request.json.get() method** is used to request the value from firebase in json format
    - Add all values in one string
    - If it get the values it will print success otherwise in exception through error
    - Create a function "***index***" that returns the (home.html) i.e. our webpage the function is mapped to the home using **'/' URL.** This means when the user navigates to **"host: 5000",** the home function will run and the output will be displayed on the webpage.

8. Use the **"render_template"** method from the flask framework, pass a HTML file to the method and it returns to the browser when the user visits the "URL" associated with that template.
    - Save the data as per latest time using **order.by** function
    - Write the IP address of the system instead of the local host, as we are dealing with two systems, ESP32 and computer. send requests to the computer by using its IP address.
    - Open Command prompt and then type **ipconfig**
        - It will show the IP address and write the same as shown below



```
def index():
    try:
        document_ref = firebase_db.collection("data")
        data = document_ref.order_by("date", direction='DESCENDING').limit(1).get()[0].to_dict()
        return render_template("/home/home.html", data=data)
    except Exception as e:

        print(str(e))
        return jsonify({
            "status": "error",
            "message": "No data in database yet!"
        }), 400

if __name__ == '__main__':
    #app.run(debug=True)
    app.run(host='192.168.0.104', port=5000)
```

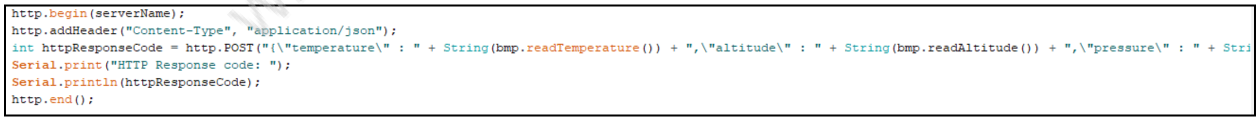
9. Open the Arduino program, which you wrote in the last class, add HTTP requests
    - Write the IP address of your computer
    - To find IP address, Go to command prompt/Terminal
    - Write ipconfig

```
Adafruit_BMP085 bmp;
String serverName = "http://192.168.0.104:5000/add-data";
```

    - Add **http** object for HTTPClient

```
WiFiClient client;
HTTPClient http;
```

    - Start the HTTP server using http **begin()**



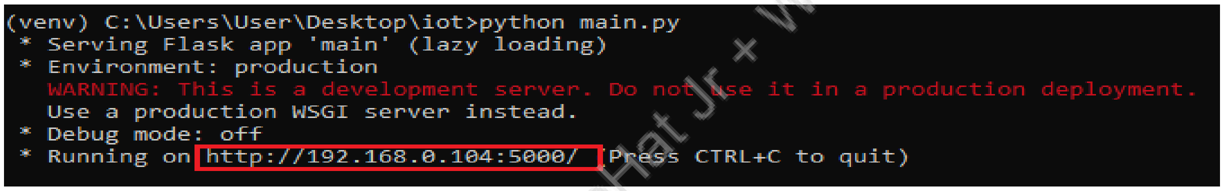
```
http.begin(serverName);
http.addHeader("Content-Type", "application/json");
int httpResponseCode = http.POST("{\"temperature\" : " + String(bmp.readTemperature()) + ",\"altitude\" : " + String(bmp.readAltitude()) + ",\"pressure\" : " + Stri
Serial.print("HTTP Response code: ");
Serial.println(httpResponseCode);
http.end();
```

    - Open your Arduino IDE,

10. **Output**
    - Compile and upload the program to ESP32 board using Arduino IDE
    - Verify the program by clicking the **Tick** option
    - Upload the program by clicking the **arrow** option
    - Run the Program
    - **Open the Google firebase in another window.**

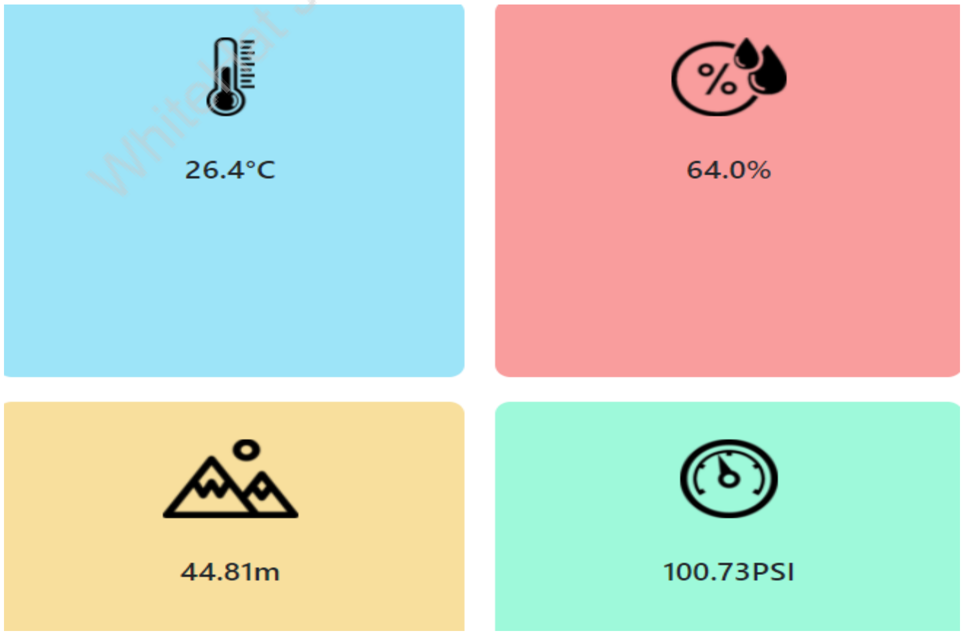
- Now run the **main.py** to run the flask server
- If get error while running the program then follow the below procedure
- Go to the folder directory and then run the below command
- Windows
  - python -m venv venv
  - venv\Scripts\activate.bat
  - pip install flask
  - pip install firebase_admin
  - python main.py

- Mac/Ubuntu -
  - python -m venv venv
  - source venv/bin/activate
  - pip install flask
  - pip install firebase_admin
  - python main.py



```
(venv) C:\Users\User\Desktop\iot>python main.py
 * Serving Flask app 'main' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://192.168.0.104:5000/ (Press CTRL+C to quit)
```

- Copy the http://192.168.0.104:5000/ and paste on the browser and click the enter button.
- Output window will be shown like this.



- We learned about RGB LED and how to fade an LED.

What's NEXT?

In the next class, we will learn **about Robotics.**

Expand Your Knowledge

To know more about **Flask framework** click here.