Topic	Weather Monitoring System - 2	
Class Description	**Student will be introduced to flask server and API through which fetch the sensor's data from Google firestore**	
Class	**PRO C252**	
Class time	**50 mins**	
Goal	• Creation of webpage • Creation of flask server	

Resources Required	• Teacher Resources:
	  ○ Laptop with internet connectivity
	  ○ Earphones with mic
	  ○ Notebook and pen
	  ○ Smartphone
	• Student Resources:
	  ○ Laptop with internet connectivity
	  ○ Earphones with mic
	  ○ Notebook and pen	

Class structure	**Warm-Up** **Student-Led Activity -1** **Student-Led Activity -2** **Wrap-Up**	**10 mins** **15 mins** **15 mins** **10 mins**
Credit & Permissions:	**Code samples used for Firebase-Google Authentication are licensed under the Apache 2.0 License.** **Expo documentation used from - https://expo.io** **Note: Keep this row section only if applicable**	
WARM-UP SESSION - 10 mins		

Teacher Action	Student Action
Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today? **Following are the WARM-UP session deliverables:** • Greet the student. • Revision of previous class activities. • Quizzes.	**ESR**: Hi, thanks! Yes, I am excited about it! Click on the slide show tab and present the slides
WARM-UP QUIZ Click on In-Class Quiz	
Activity Details **Following are the session deliverables:** • Appreciate the student. • Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.	
TEACHER-LED ACTIVITY-1 - 15mins	
Student Initiates Screen Share	
• **Creation of HTML page**	
Teacher Action	**Student Action**
So in the last class, we set up our database and we also get the values from the sensor. Any doubts from the last class *In case of doubt, clarify the doubts of the student*	

Note: While downloading the reference file. please change key.py as per teacher/student firebase credentials and please set google firebase complete set up and change your IP address as per system So today we will 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 will do little changes in our Arduino programm to make this happen. Let's start So we are familiar with the flask server as we have done in the networking module too. *Let's set up connections first* *Note: The connections are the same as we did in the last class, so if the old setup is ready, then use it, if not, help the student to set it up again.*	
Step -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	
Step -2: Let's do connections: ● Supply VCC(positive) from ESP32 (VIN PIN) to the breadboard positive rail.	

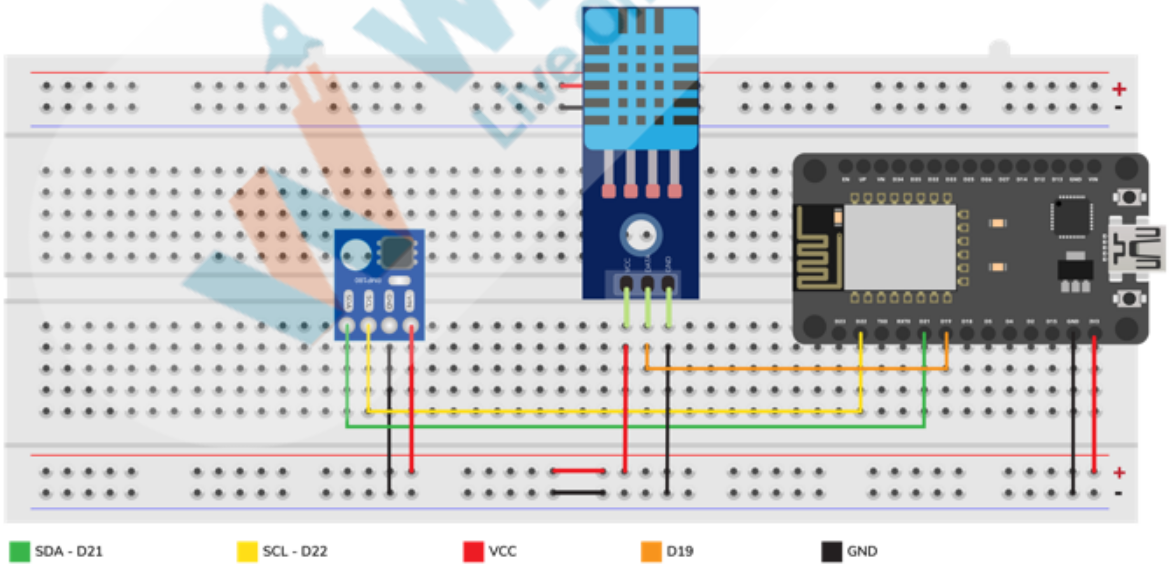
- Supply GND(negative) from ESP32 (GND PIN) to breadboard negative rail.

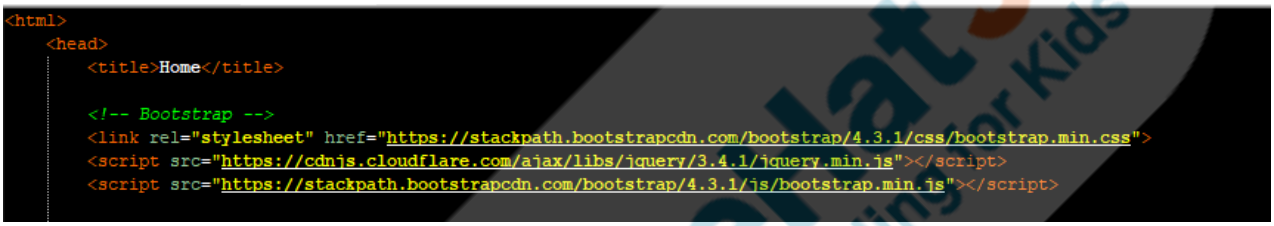
Connect BMP180 sensor

- 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

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



So Let's design our Wheather Monitoring webpage using HTML and then, later on, we set up the server.	
Can you tell me what is the use of bootstrap? Excellent! In Bootstrap we have precompiled style and setting files that are quick and easy to use for any web development styling and moreover, it makes your website responsive	ESR: To make the webpage responsive
 	
Bootstrap follows a box model, and works in **rows** and **columns**. This means that everything that our page consists of is made up of **rows** and **columns.** One thing, however, to always keep in mind while working with bootstrap is that the content should always be inside a **column** instead of directly being inside a **row.** But before that, we need to add style first to our webpage In the style tag, we will add width, height, background color for our 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>
```

Now let's do the bootstrap part

Here, we can see that we first have a ** ** tag, which 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**

Now, what does all these mean?

n bootstrap, 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**. Therefore here, ***col-sm-12*** means to have the full width of the row on the small screen, ***col-md-12*** means to have the full width of the row on medium screen and ***col-lg-12*** means to have the full width of the row on a large screen. Add headers for temperature, pressure, altitude and humidity.	

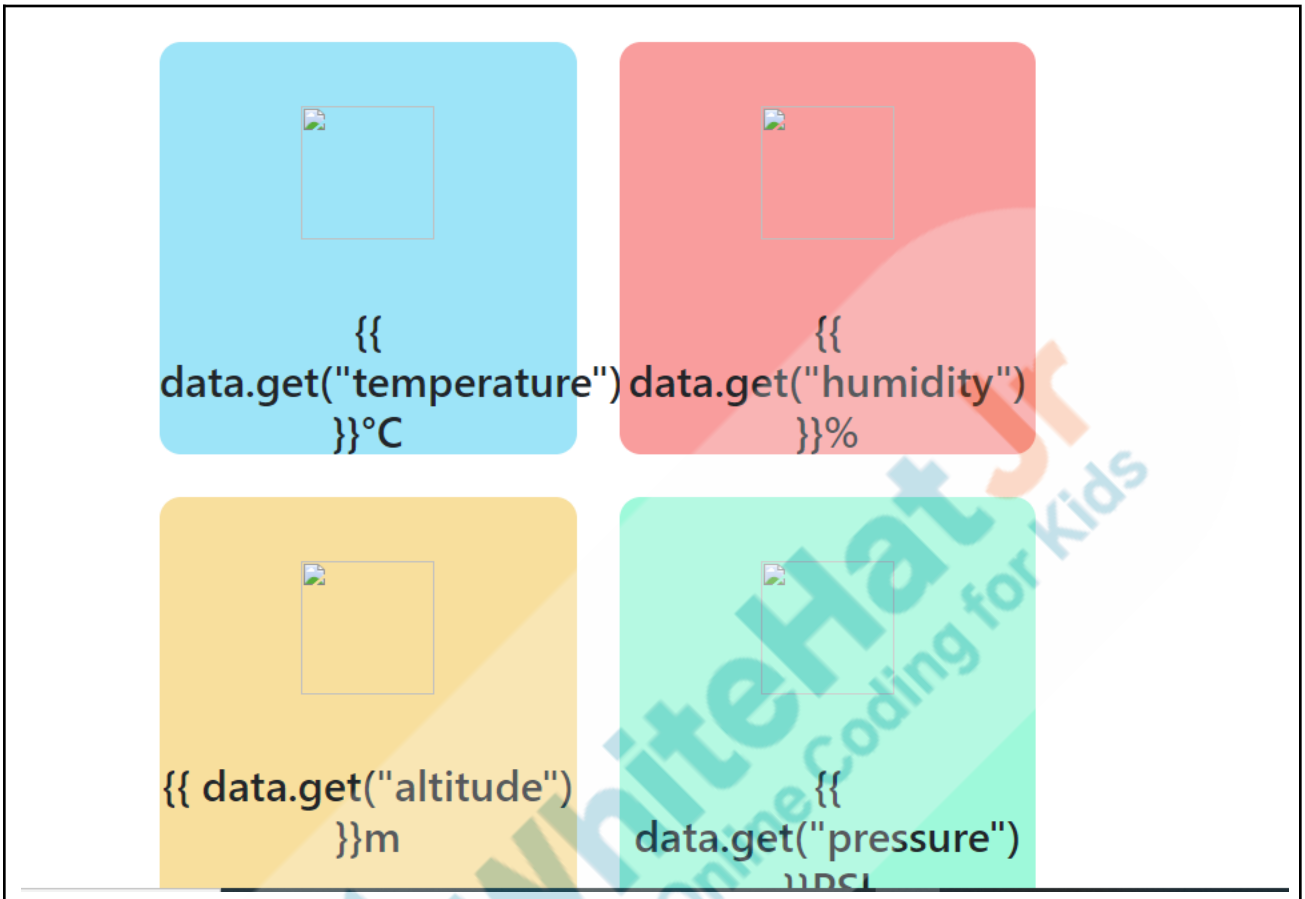
```
</head>
<body>
    <div class="row">
        <div class="col-sm-12 col-md-12 col-lg-12 m-3">
            <img src="https://s3-cdnwhjr.whjr.online/website/desktop/logo_whjr.png" height="80em" />
        </div>
    </div>
    <div class="row">
        <div class="col-sm-12 col-md-12 col-lg-12 m-3 text-center">
            <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

Note: data.get() alue name like temperature, humidity, pressure, altitude should be exactly same as written in Google firebase while entering fields.

```
        <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>
```

So our design page is done . If you read your HTML file it will look like this.



So our design is ready, but we cant see actual values, So let's set up the server and fetch the real-time values through API	
Student Stops Screen Share	
So it's time to set up the server. Please share your screen with me. Let's try. I will guide you through it.	
STUDENT-LED ACTIVITY-2 - 15 mins	
• Ask the student to press the ESC key to come back to the panel. • Guide the student to start Screen Share. • The teacher gets into Full Screen.	

Student Initiates Screen Share	
ACTIVITY • Creation of flask server	
Teacher Action	Student Action
Teacher guides the student to download boiler plate	*Student clicks on student Activity 1*
Let's set up the server Flask is the python module used to create instances of web applications Even Flask, we have used in many previous applications too, As we have seen, all of the applications that we use to run on "localhost:5000" So first of all we will build a Flask application that accepts either query strings, form data, or JSON objects that later on will return HTML Pages Using the ***"render_template"*** method from the flask framework, we passed an HTML file to the method and it returned 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 *Note: This part is on the boiler plate*	

```
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: key.py is the file where we will save creds of Google firebase

Note: Student start writing code from here

Note: Download the boiler plate code and open the key.py, then save all the Google Firestore credentials we downloaded during the generation of the new key in the last class. Paste under the creds as shown in picture.

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
}
```

Next we need to install Flask on our system if it's not installed in the system

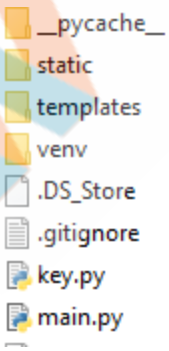
To avoid conflicts with libraries, install Flask in a virtual environment.

To create and activate a virtual environment, we can simply run the following commands -

Mac/Ubuntu -

python -m venv venv

!

source venv/bin/activate **Windows -** ***python -m venv venv*** ***venv\Scripts\activate.bat*** Next, we will run the following command to install ***flask*** into our virtual environment ***pip install flask*** Next, we will 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***. So, folder called "***templates***" contain HTML page there. Here is how the web app directory tree should be like at this point: Python or .py script stays outside of the templates folder. So first will start working on our main.py file	
	
Also, to start the application, we would 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"])
```

Let's make function add_data()

- Make variable to save values fo **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 get the values print success otherwise in exception through error

```
def add_data():
    try:
        temperature = request.json.get("temperature")
        humidity = request.json.get("humidity")
        altitude = request.json.get("altitude")
        pressure = request.json.get("pressure")
        document_ref = firebase_db.collection("data")
        add_values = document_ref.document().create(dict(temperature=temperature, humidity=humidity, altitude=altitude, pressure=pressure, date=datetime.datetime.utcnow()))
        return jsonify({
            "status": "success"
        }), 201
    except Exception as e:
        return jsonify({
            "status": "error",
            "message": str(e)
        }), 400

@app.route("/")
```

Next, we are creating 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.

Using the ***"render_template"*** method from the flask framework, we passed an HTML file to the method and it returned to the browser when the user visits the "URL" associated with that template.

Save the data as per latest time using **order.by** function

However, here we need to write the IP address of the system instead of the local host, as we are dealing with two systems, ESP32 and your computer. So, we 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

Note: Write system's **IP address** in the marked filed

```
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)
```

Now open the Arduino program, which we have written in the last class, Now we will add some HTTP requests

Write the IP address of your computer

To find IP address, Go to command prompt/Terminal

Write ipconfig

Earlier it was like this:

```
Adafruit_BMP085 bmp;
```

Now it will be like this:

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

Add **http** object for HTTPClient

Earlier it was like this:

```
WiFiClient client;
```

Now it will be like this:

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

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

Earlier it was like this:

```
if (isnan(h))
{
  Serial.println("Failed to read from DHT sensor!");
  return;
}
else
{
  Serial.print("Humidity: ");
  Serial.print(h);
}
```

Now it will be like this:

```
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,

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 too 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

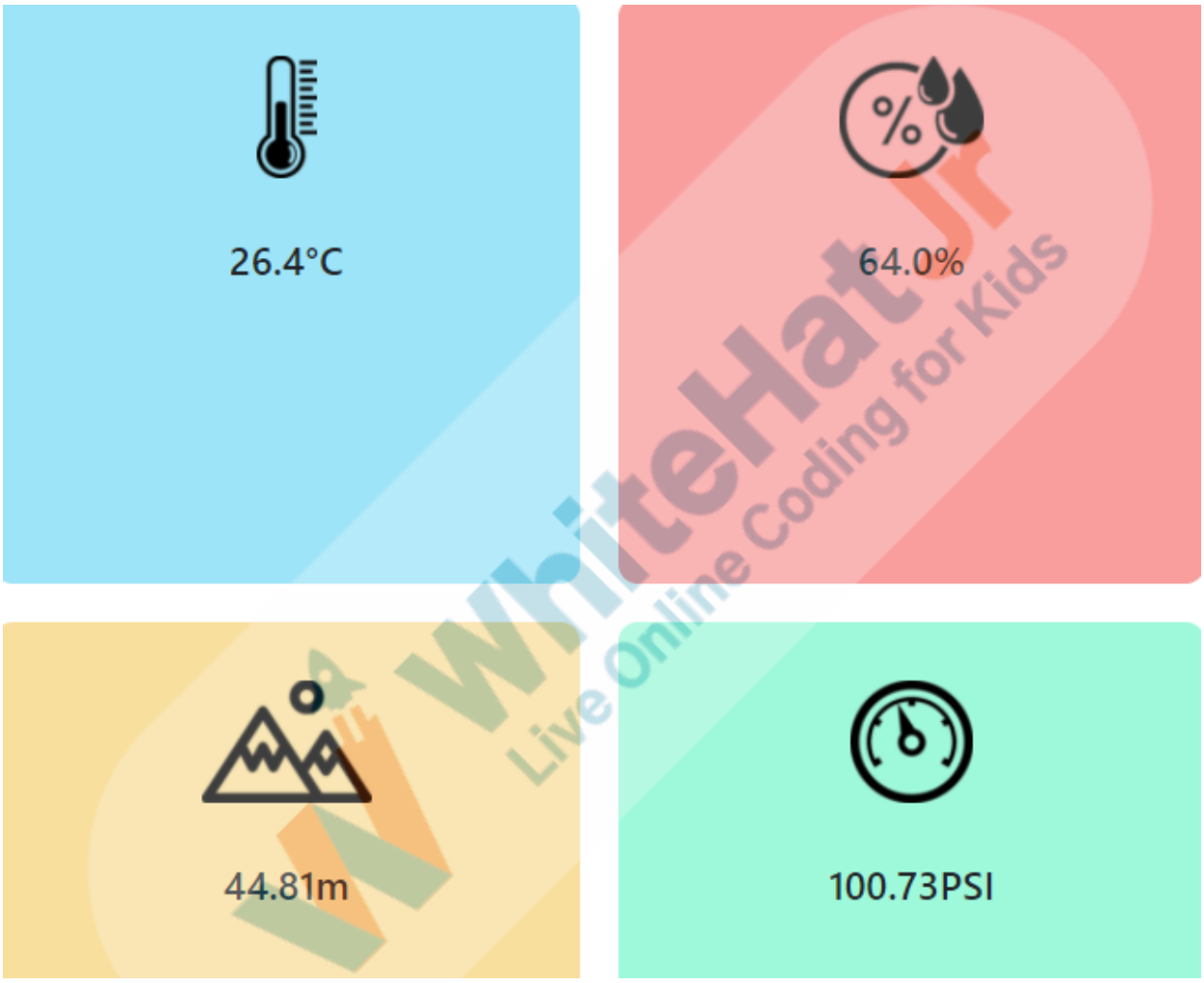
python -m venv venv
venv\Scripts\activate.bat
pip install flask
pip install firebase_admin
python main.py

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)
```

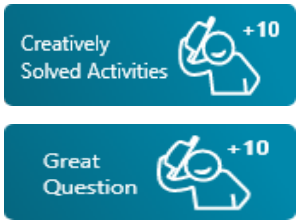
- Copy the http://192.168.0.104:5000/ and paste on the browser and click the enter button.
- Output window will be shown like this.



So, today we learned about RGB LED and how to fade an LED.

That's fun!

Teacher Guides Student to Stop Screen Share

WRAP-UP SESSION - 05 mins	
Activity details **Following are the WRAP-UP session deliverables:** • Appreciate the student. • Revise the current class activities. • Discuss the quizzes.	
WRAP-UP QUIZ Click on In-Class Quiz	
Activity Details **Following are the session deliverables:** • Explain the facts and trivia • Next class challenge • Project for the day • Additional Activity (Optional)	
FEEDBACK • **Appreciate and compliment the student for trying to learn a difficult concept.** • **Get to know how they are feeling after the session.** • **Review and check their understanding.**	
Teacher Action	**Student Action**
You get “hats-off” for your excellent work! In the next class, we will learn about Robotics	*Make sure you have given at least 2 hats-off during the class for:* 

	
PROJECT OVERVIEW DISCUSSION Refer the document below in Activity Links Sections	
Teacher Clicks ✖ End Class	
ADDITIONAL ACTIVITIES (Optional)	
Additional Activities	

ACTIVITY LINKS		
Activity Name	**Description**	**Links**
Teacher Activity 1	Reference Code -	https://github.com/procodingclass/Pro-C252-Reference-Code
Teacher Reference 1	Project	https://s3-whjr-curriculum-uploads.whjr.online/3a4b8576-1d5c-4f56-9949-2b92e47360d7.docx
Teacher Reference 2	Project Solution	https://github.com/procodingclass/PRO-252-Project-Solution.git

Teacher Reference 4	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/96b3f5b5-b826-435f-8b8a-d9bacbdeb359.docx
Student Activity 1	Student Boiler Plate	https://github.com/procodingclass/Pro-C252-Student-BoilerPlateCode