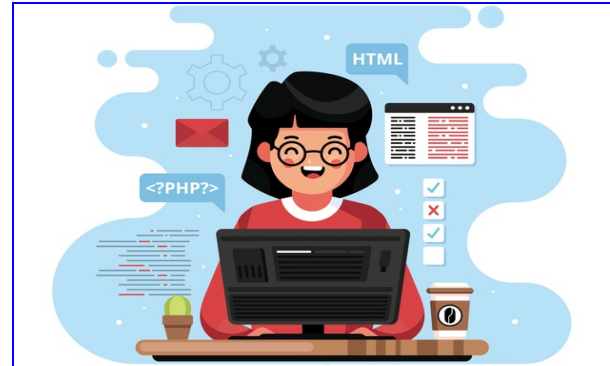**WhiteHat Jr**
Live Online Coding for Kids

Weather Monitoring System - 1

What is our GOAL for this CLASS?
In this class, we were introduced to **no-SQL databases** using Google Firestore and we set up our own **weather monitoring system.**

What did we ACHIEVE in the class TODAY?

- We were introduced to **Firestore.**

- We learned **Weather Monitoring set up & Programming**

Which CONCEPTS/ CODING BLOCKS did we cover today?

- Wrote a program on **Arduino IDE** to get value from sensors

- We Set up "**No SQL database**" using Google Firestore

- We Created an **HTML page** to display the sensor's value

- Set up

- Set up **of web server** using flask.

- We learned to send values from the **sensor to ESP32**, ESP32 will save data to **firestore**, and from **firestore using API** we fetched data to a flask server, and from the server will displayed values on an HTML page.

How did we DO the activities?

1. Set up the database
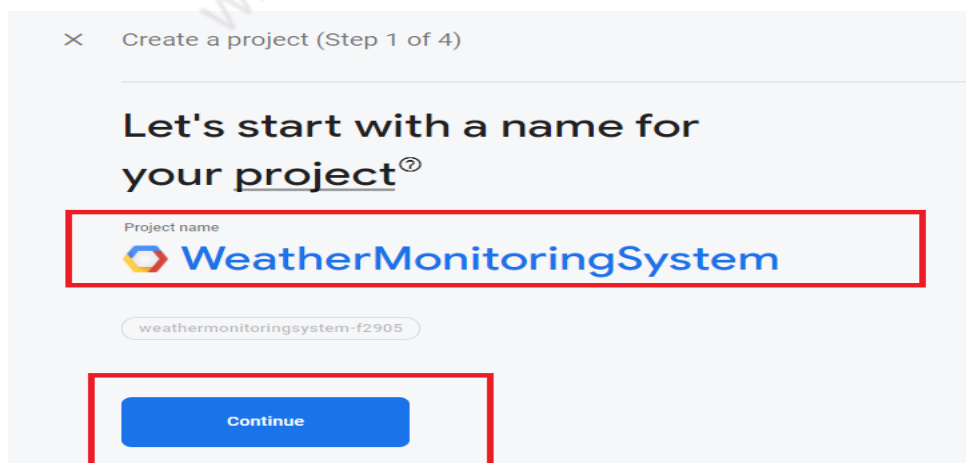   - Open the link and click on sign in
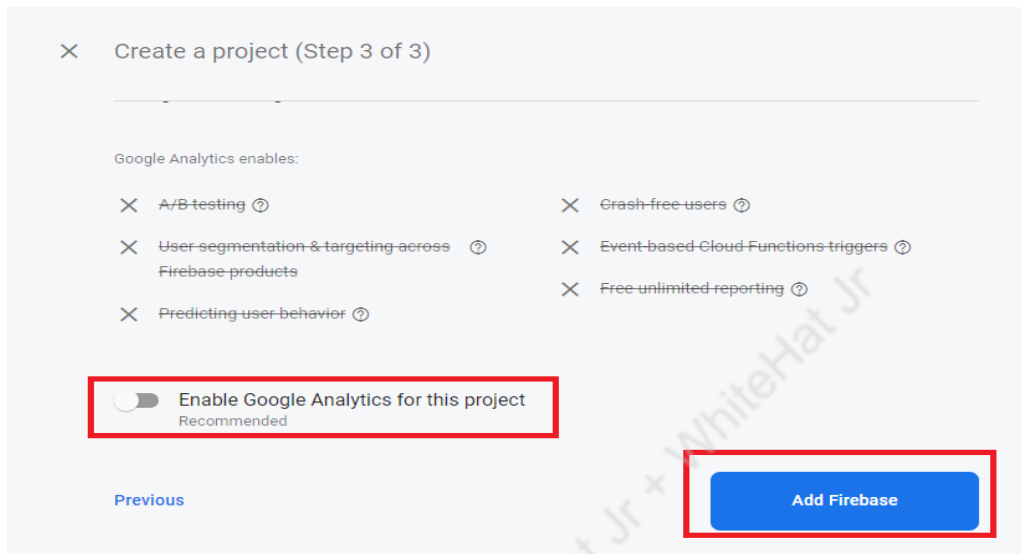
- Click on **Go to Console**
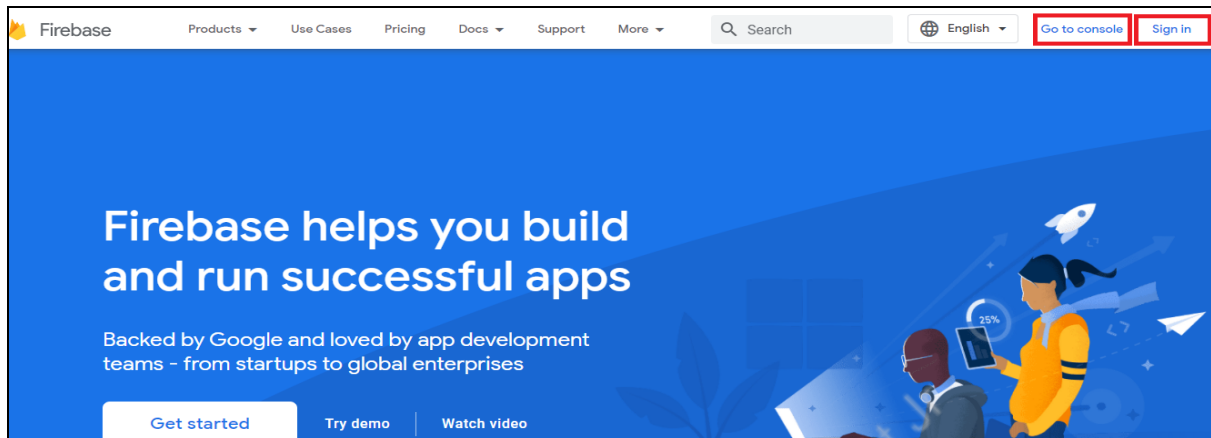
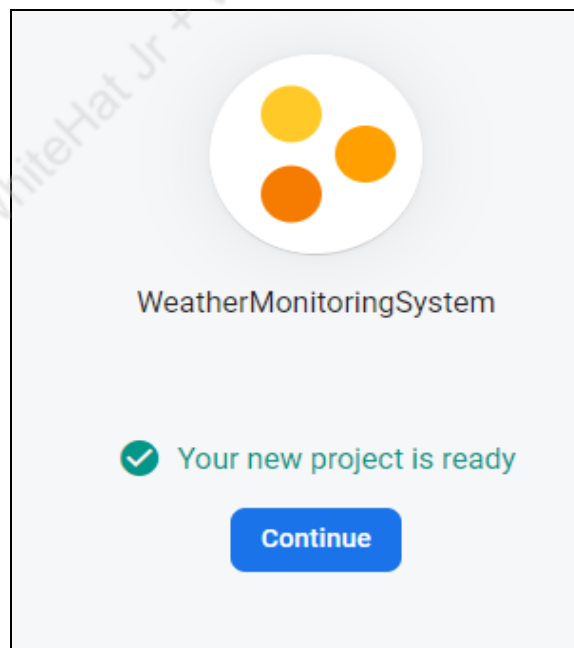- Select the "**Add project"** option to create a new project.

- **Enter your project name**, give the project a suitable name, and then click the blue **Continue** button.
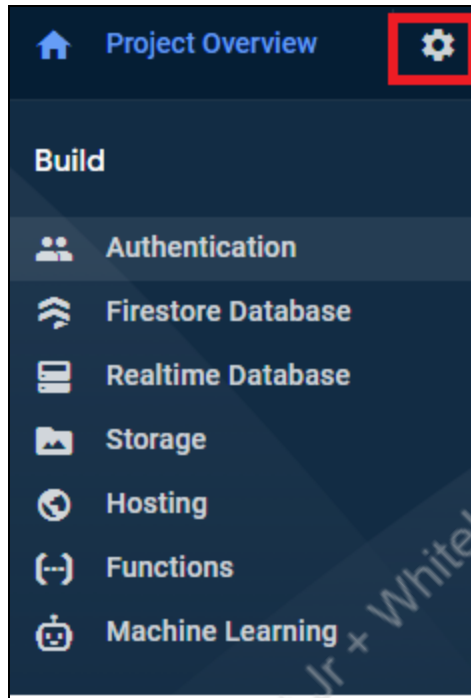
- Click on **Continue** button.
- Disable the "**Enable Google Analytics"** for this project option as it will not be needed.
- select **Add Firebase**

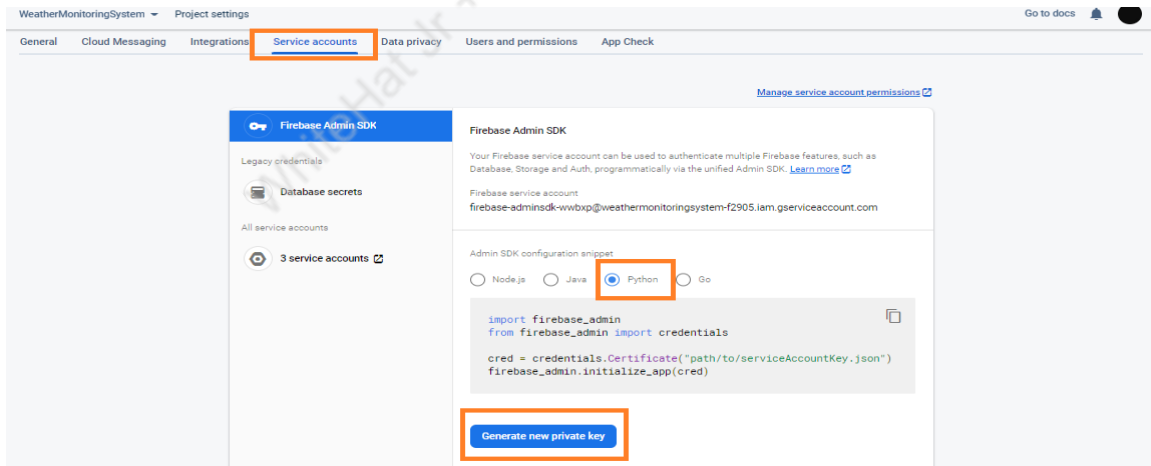- Firebase will begin set up the project. Once completed select **Continue**

- Click **"Settings Symbol"** next to **Project Overview**

- Select **"Service Accounts"** and the select "**Python**" and then click on **"Generate new private key"**

- Go to **Downloads** and save this file.
- Use this file while setting up our server. Save this.

● Click on "**Firestore Database"**
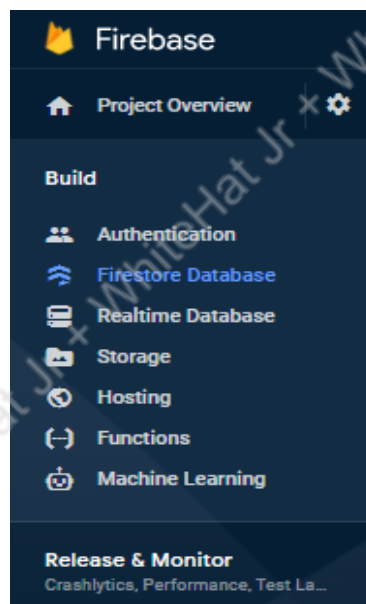
● Select "**Create database"**
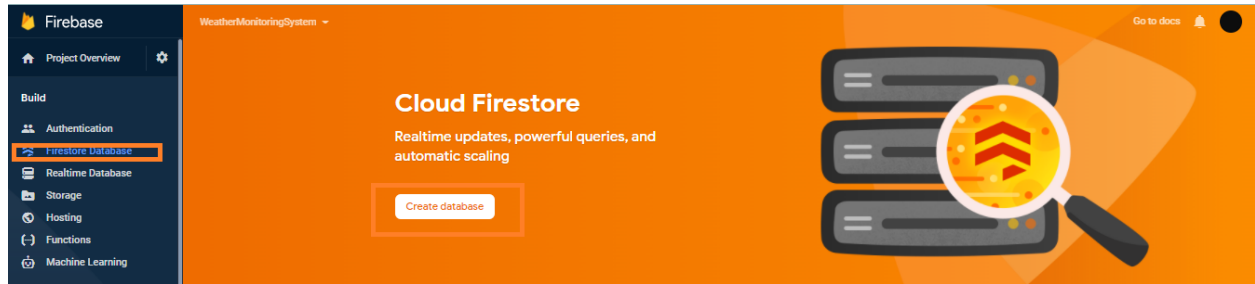
- Select **"Start in test mode", Click on "Next"**

- Select "Asia" cloud Fire store station, it can be chosen any Asian direction
- Click on **"Enable"**

- Click on "**+Start collection**"
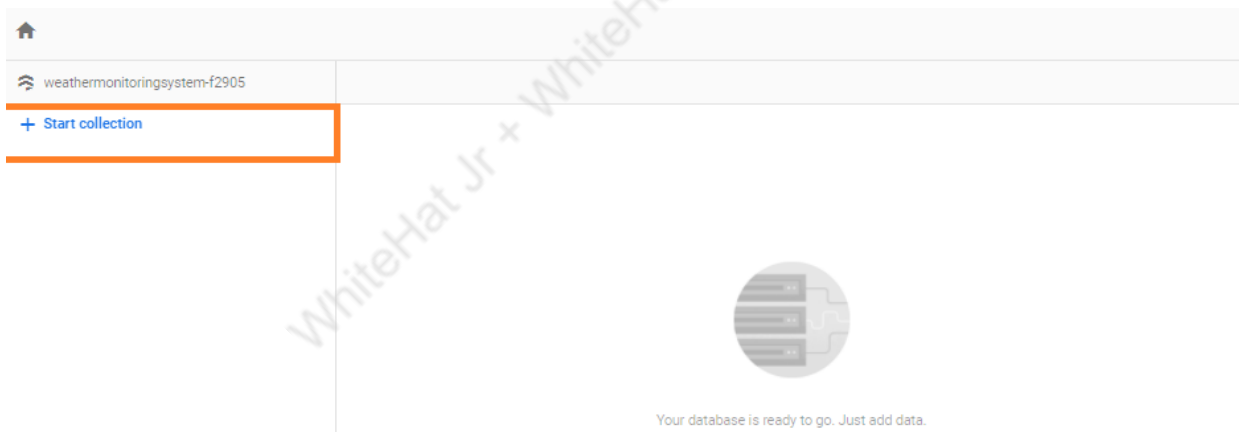
- Write the **"Collection ID" data**
- *Click on "**Next**"*

Start a collection

1 Give the collection an ID ———— 2 Add its first document

Parent path

/

Collection ID ⓘ

data

Cancel   **Next**

- Write the **DocumentID** data and insert the default values, of t**emperature, humidity, pressure, altitude, and time**.
- As per input add data types. For a time it would be **timestamp** and for others, it would be **numerical.**
- Enter default values in **field**

/data

Document ID ⑦

data

Field	Type	Value	
temperature	= number ▾	25	⊖

Field	Type	Value	
humidity	= number ▾	64	⊖

Field	Type	Value	
pressure	= number ▾	84	⊖

Field	Type	Value	
altitude	= number ▾	29.68	⊖

Field	Type	
Date	= timestamp ▾	⊖

Date

Feb 3, 2022 📅

Time

00:00:00000

- firestore database is set now, but to insert the values in database call API but before that get values from sensors.
- store the next task is to get the values from the ESP32

2. **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

3. **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.

4. **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

5. **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

■ SDA - D21   ■ SCL - D22   ■ VCC   ■ D19   ■ GND
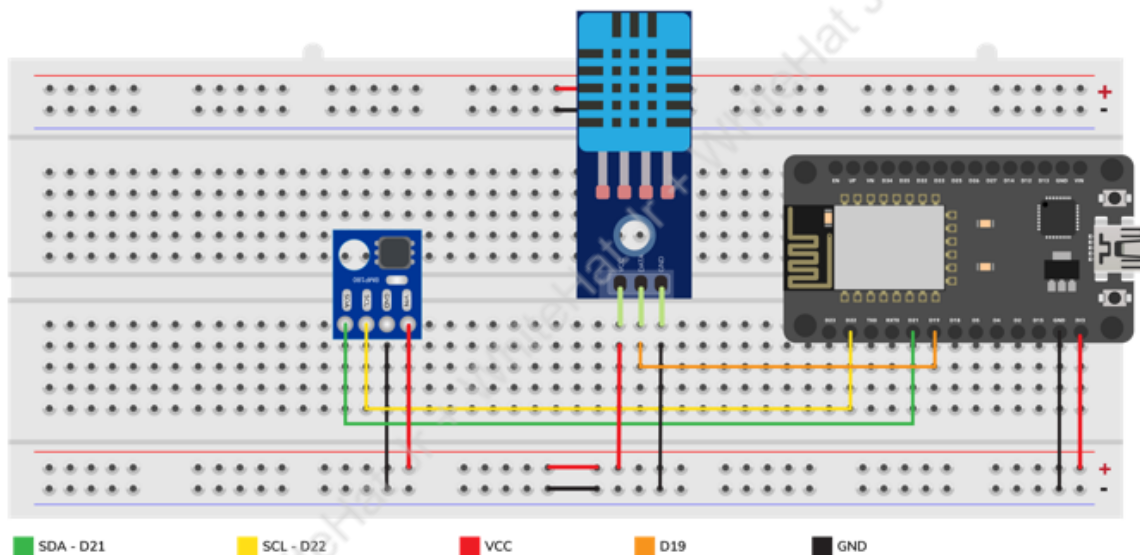
6. **Include libraries**
   ● The top of the program has includes. write all supporting header files and libraries
   ● **include keyword** is used to import libraries in embedded language as we used to import in python language
   ● **WiFi.h:** WiFi library will be able to answer all HTTP request
   ● **Adafruit BMP085.h** This library supports the **pressure** sensor **BMP180**
   ● **DHTTYPE DHT11:** This library supports the temperature and Humidity sensor DHT11

```
#include <Wire.h>
#include <Adafruit_BMP085.h>
#include <WiFi.h>
#include <HTTPClient.h>
#include "DHT.h"
#define DHTPIN 19
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
```

- connect with ESP32 with the WiFi. For that, use SSID(Wi-Fi credentials i.e WiFi name and WiFi Password)
- **WLAN_SSID, WLAN_PASS** are the variables that are used to save WiFi credentials.
- Set the **SSID** and **password**
- Create object **bmp** for **Adafruit_BMP085**

```
#define WLAN_SSID        "WR3005N3-757E"
#define WLAN_PASS        "70029949"
```

7. **Initialize the setup()**
   - **Serial. begin(9600)** is used for data exchange speed.
   - **Serial. println** is used to print the statement
   - **bmp.begin()** is used to begin the process
   - **Serial.println** used to print data. Print ("Could **not found",** if its fail to begin the process)

```
void setup() {
  Serial.begin(9600);

  Serial.print("Connecting to ");
  Serial.println(WLAN_SSID);

  WiFi.begin(WLAN_SSID, WLAN_PASS);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());

  if (!bmp.begin()) {
    Serial.println("Could not find a valid BMP085/BMP180 sensor, check wiring!");
    while (1) {}
  }
  dht.begin();
}
```

8. To execute the main process write the **void loop()**
   - **Serial.print is** used to print data
   - **readTemperature()** will read the temperature of the surroundings amd print the result in **celcius**
   - **readPressure()** will read the pressure of the surroundings print the result in **pascal**
   - **readAltitude()** will read the Altitude value of the surroundings print the result in **meters**
   - Set the **delay** of **500 ms**

```
void loop() {

  if (WiFi.status() == WL_CONNECTED) {
    WiFiClient client;
    Serial.print("Temperature = ");
    Serial.print(bmp.readTemperature());
    Serial.println(" *C");

    Serial.print("Pressure = ");
    Serial.print(bmp.readPressure());
    Serial.println(" Pa");

    Serial.print("Altitude = ");
    Serial.print(bmp.readAltitude());
    Serial.println(" meters");
```

   - **Create float h** variable to store decimal value
   - **readHumidity()** will read the humidity of the surroundings
   - isnan() function is used to return a null value. Check **if** any reads failed **then exit** using **isnan() function**
   - Print the humidity value.
   - Set a **delay** of **5000** ms

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

  delay(5000);
}
```

9. **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

```
23:46:50.472 -> Temperature = 25.40 *C
23:46:50.519 -> Pressure = 100982 Pa
23:46:50.519 -> Altitude = 28.68 meters
23:46:50.566 -> Humidity: 73.00Temperature = 25.40 *C
```

   ● So we got our values, now next things is to set up server and call the API

What's NEXT?
In the next class, we will learn about flask servers and how to display value on HTML page

Expand Your Knowledge
To know more about **Database** click here.