Topic	WEATHER MONITORING SYSTEM - 1	
Class Description	**Students will be introduced to no-SQL databases using Google Firestore. They will also set up their own weather monitoring system.**	
Class	**PRO C251**	
Class time	**50 mins**	
Goal	• Introduction to Firestore • Weather Monitoring set up & Programming	

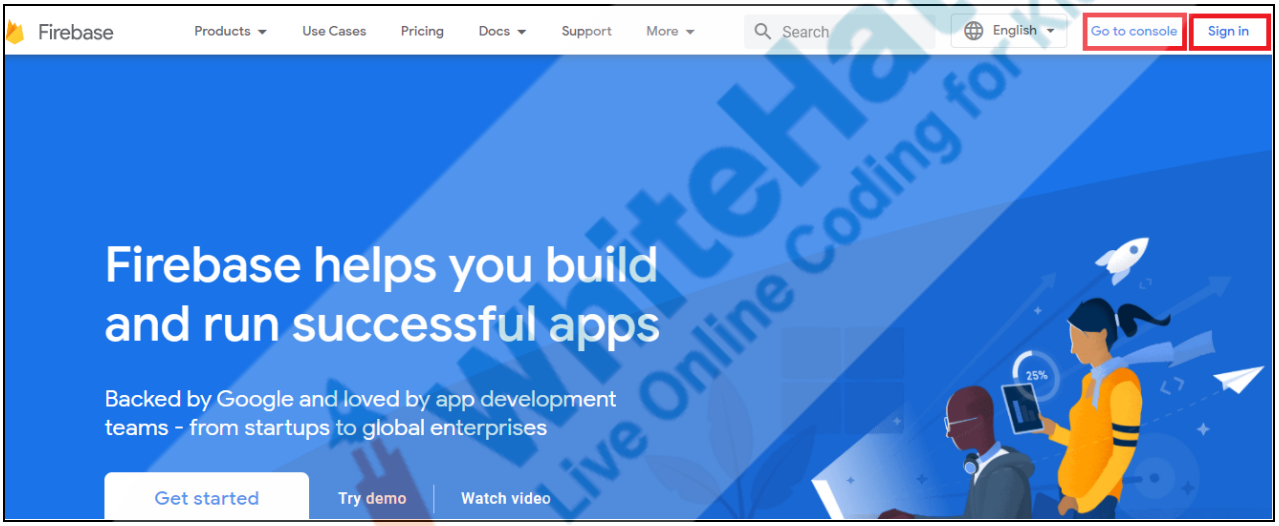
Resources Required	• Teacher Resources:
	  ○ Laptop with internet connectivity
	  ○ Earphones with mic
	  ○ Notebook and pen
	  ○ Smartphone
	• Student Resources:
	  ○ Laptop with internet connectivity
	  ○ Earphones with mic
	  ○ Notebook and pen	

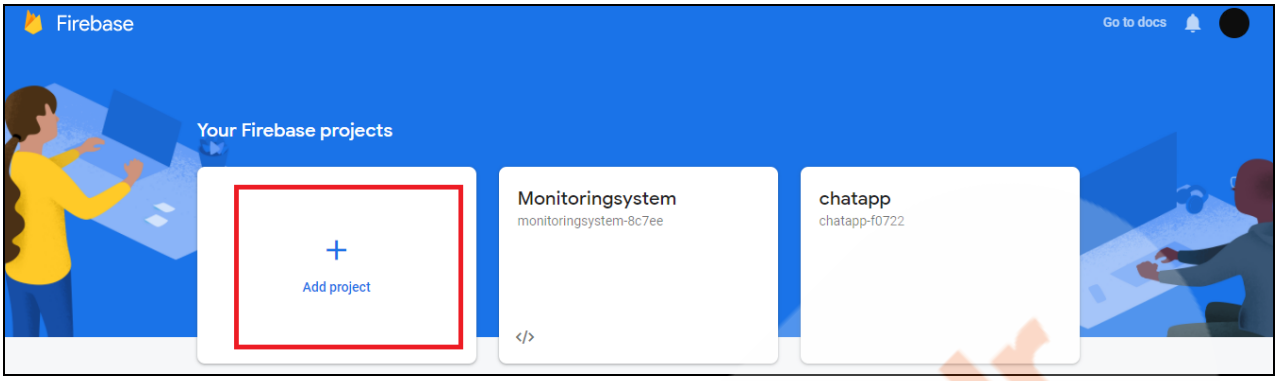
Class structure	**Warm-Up** **Teacher-Led Activity** **Student-Led Activity** **Wrap-Up**	**10 mins** **15 mins** **15 mins** **10 mins**
Credit & Permissions:	**Code samples used for Firebase-Google**	

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	
Teacher Initiates Screen Share	
ACTIVITY • **Introduction to Firestore**	
Teacher Action	Student Action
How was your experience with the IoT module? Currently you are learning how to make devices smart by using the internet, cloud servers and controller ESP32.	**ESR:** Varied!

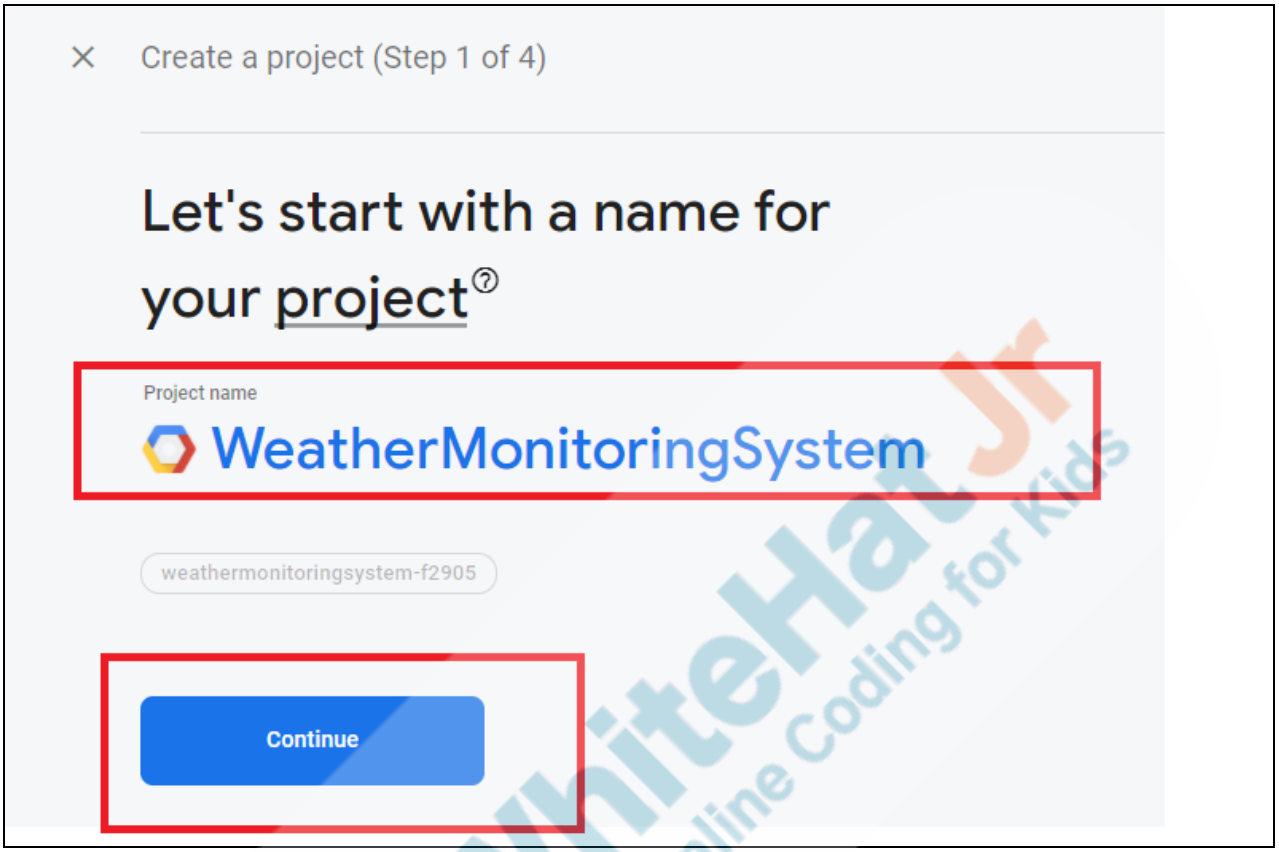
With the knowledge that we have learnt in the IOT module, we will move forward with our next module - Robotics. We will learn more about electronics circuits in this module and learn robotics. Don't you think it would be awesome if we can use our own server, own database so that we can make our individual weather monitoring system. This system will tell everything about your surroundings like altitude, pressure, humidity, temperature. Let's start! But can you tell me how we can make this all happen? *Note: Let' students think about this!* To make this happen we need to do a lot of things on our end 1. Write a program on Arduino IDE to get value from sensors 2. Set up "No SQL database" using Google Firestore 3. Create an HTML page to display the sensor's value 4. Set up 5. Set up a web server using flask. **Flow**: Send values from the sensor to ESP32, ESP32 will save data to firestore, and from firestore using API we will fetch data to a flask server, and from the server will display values on an HTML page.	

So, Let's set up the database first and then we will get the value from ESP32.	
Teacher clicks on Teacher Activity1	Students click on Student Activity 1
Click on **Sign in.** After Sign up, 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.

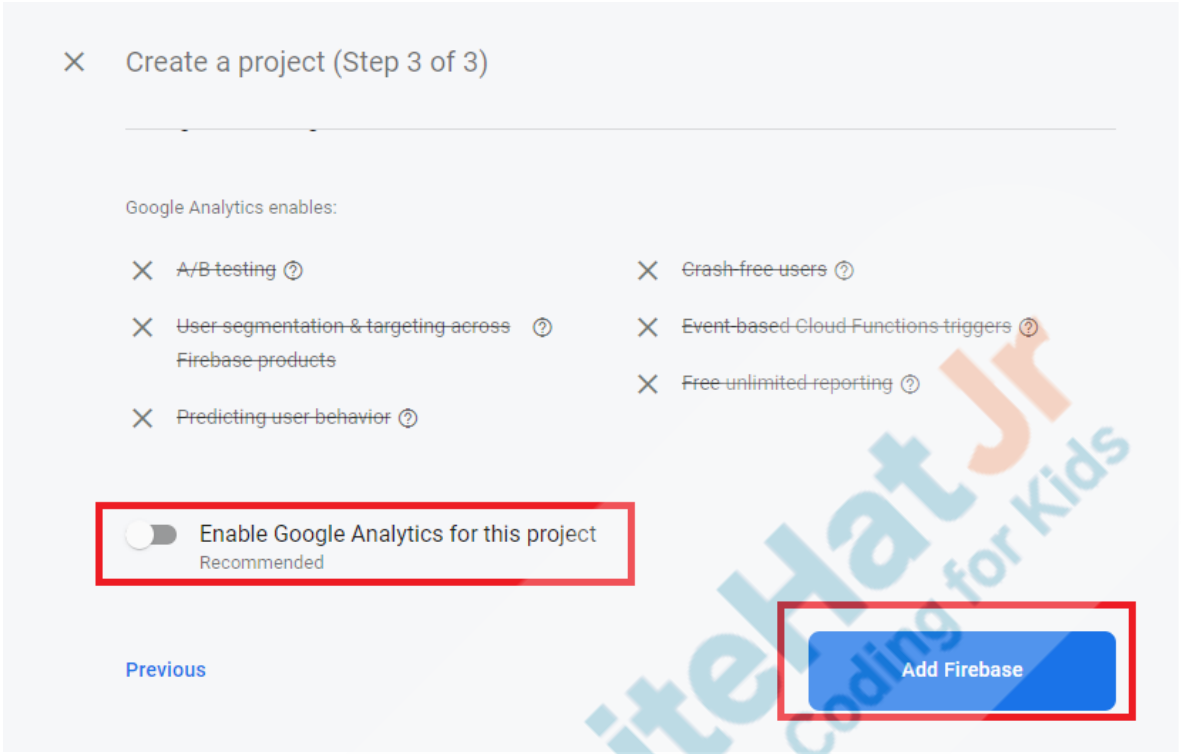
Note: Here have entered the name "WeatherMonitoringSystem", Students can enter any name but it should be without space.



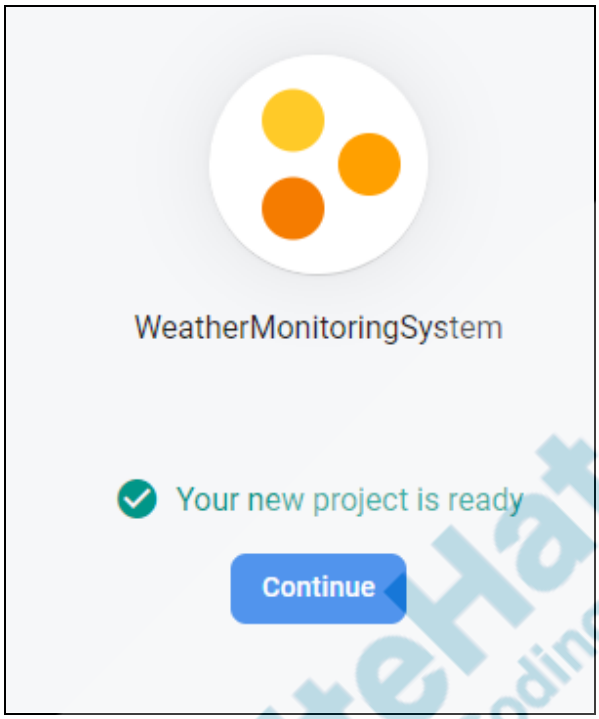
After that click on the **Continue** button again.

Disable the "**Enable Google Analytics"** option for this project as it will not be needed.

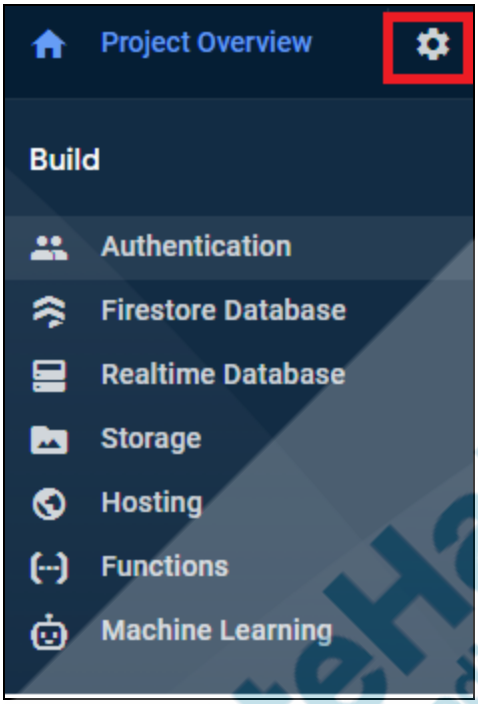
Finally, select **Add Firebase.**



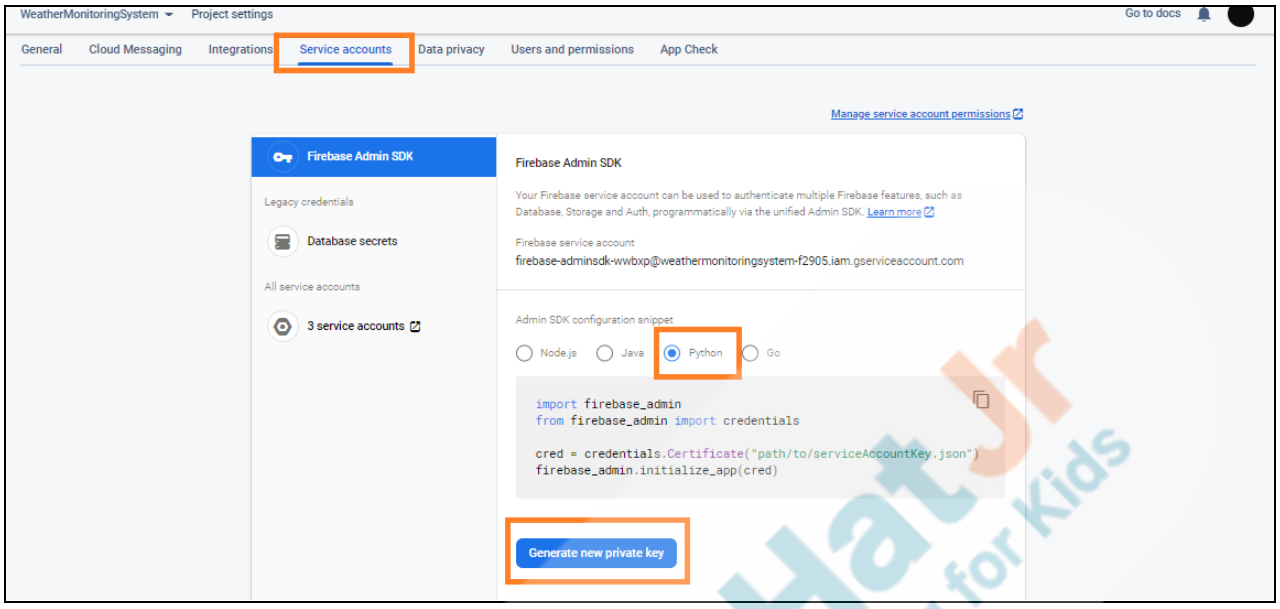
Firebase will begin to set up the project. Once completed select **Continue.**	



Click **“Settings Symbol”** next to **Project Overview.**

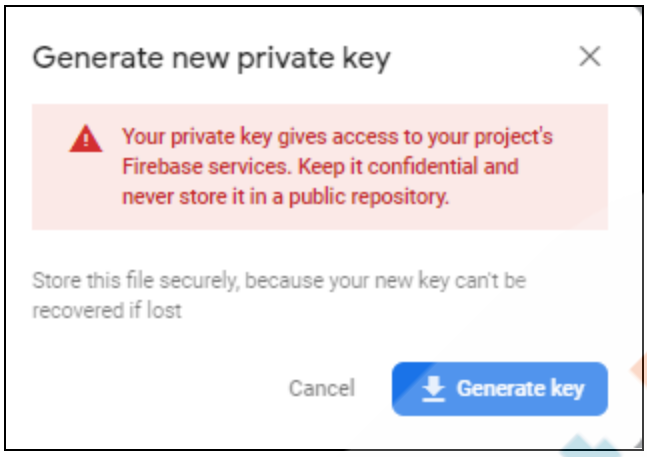


Select **“Service Accounts”** and the select “**Python”** and then click on **“Generate new private key”.**	

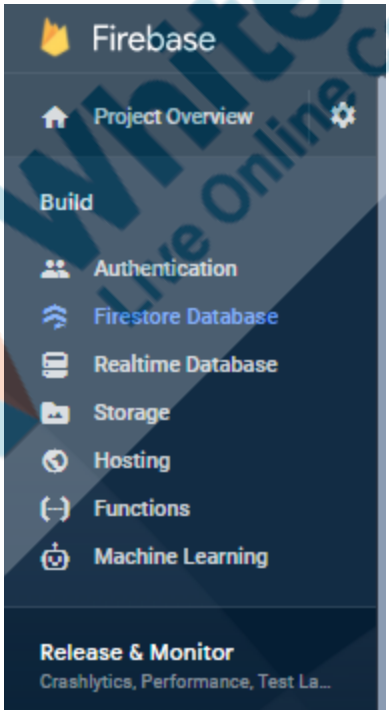


*Go to **Downloads** and save this file.*

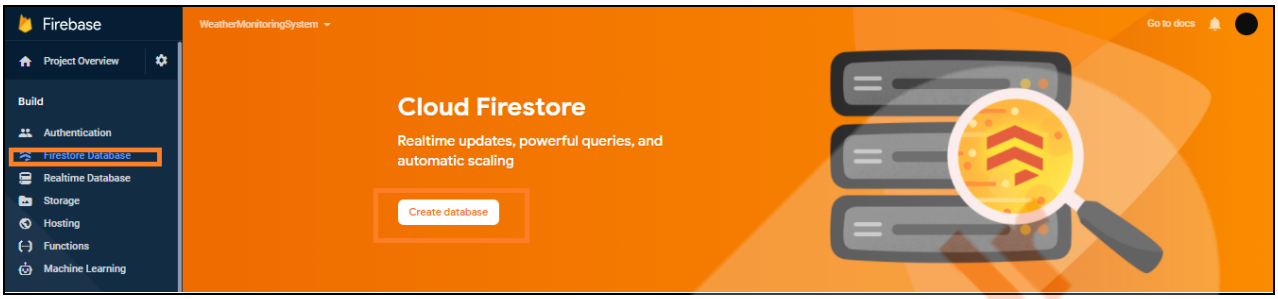
Note: We will use this file while setting up our server.
Teacher guides the student to save this file.



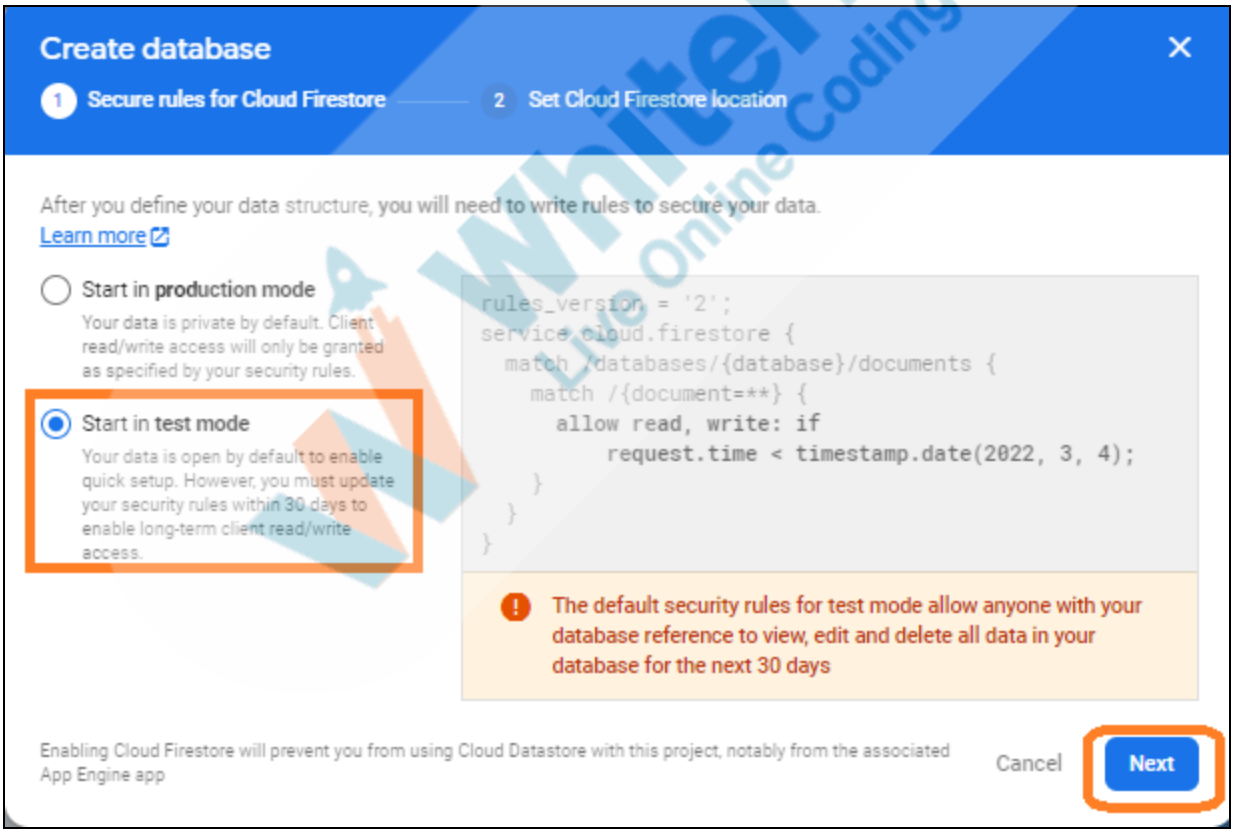
Click on “**Firestore Database**”

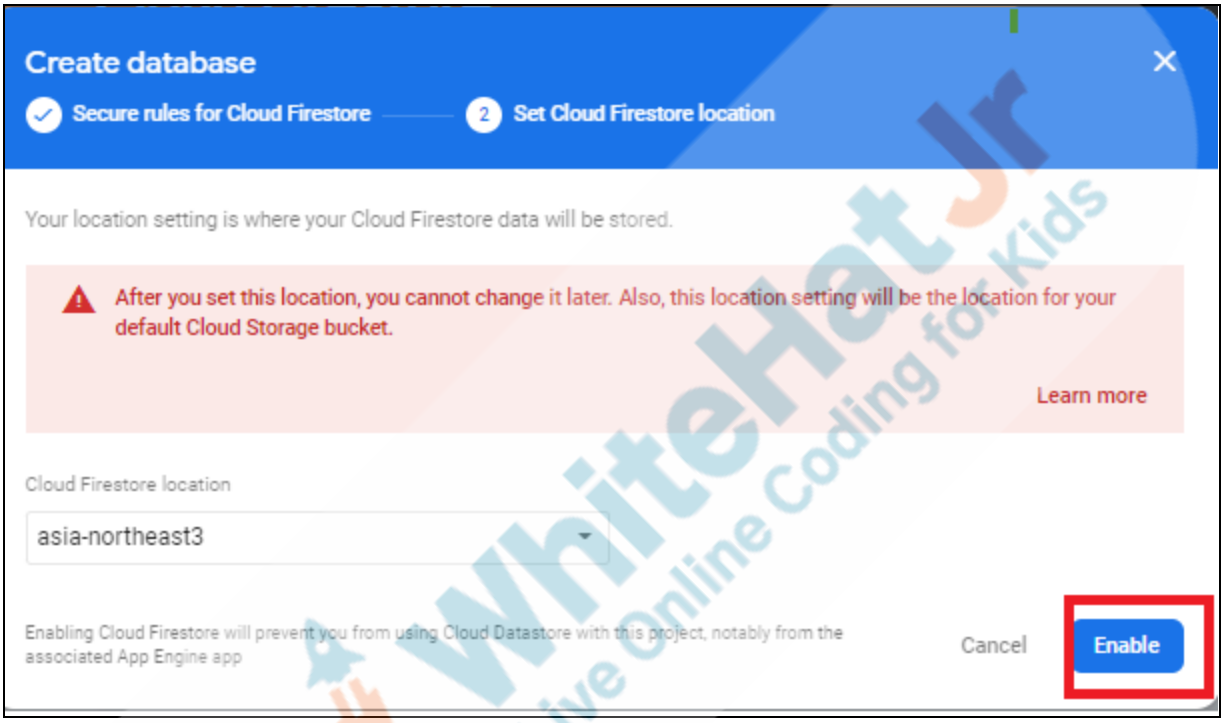


Select “**Create database”.**



Select **“Sart in test mode”, Click on “Next”.**

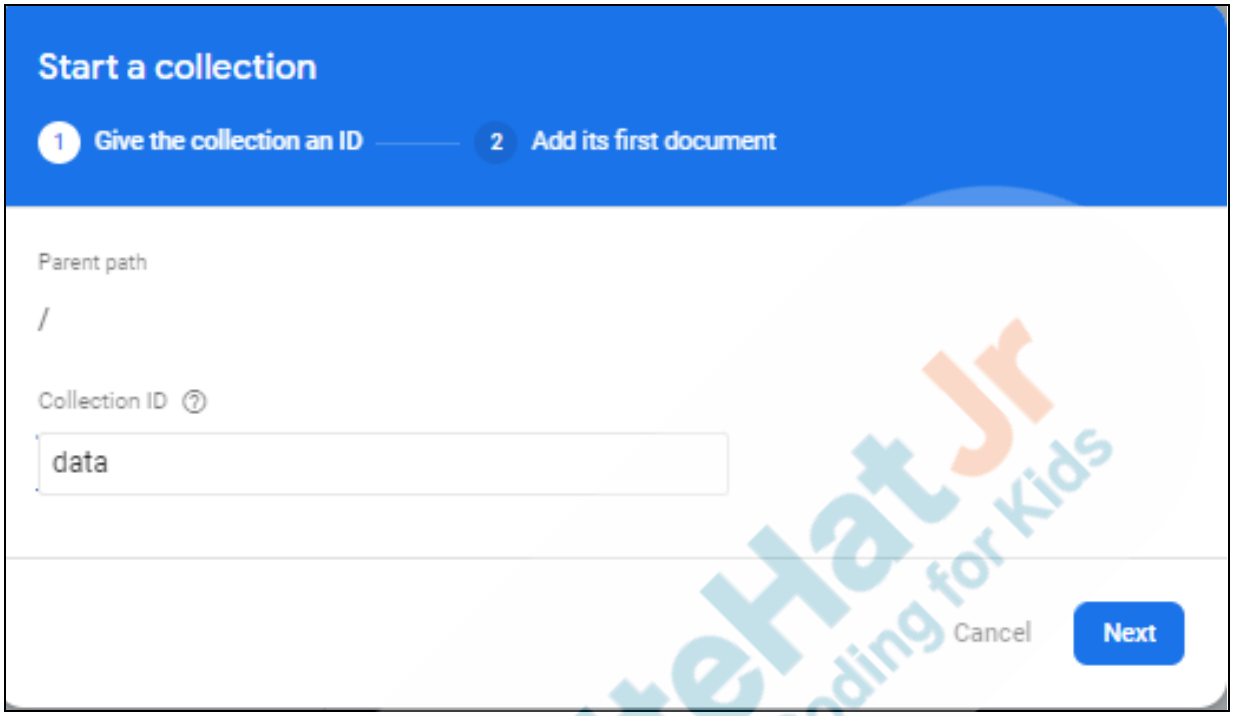


Select “Asia” under Cloud Firestore location. You can choose any location. Click on **“Enable”.**		
		
Click on “**+Start collection”.**		

Write the **“Collection ID” data.**

Note: Write data only as this would be our API name

*Click on “**Next”.***



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

So our firestore database is set now. To insert the values in the database, we need to call an API. But before that we must get values from sensors.

Teacher Stop Screen Share

STUDENT-LED ACTIVITY-1 - 15mins

- Ask the student to press the ESC key to come back to the panel.
- Guide the student to start Screen Share.

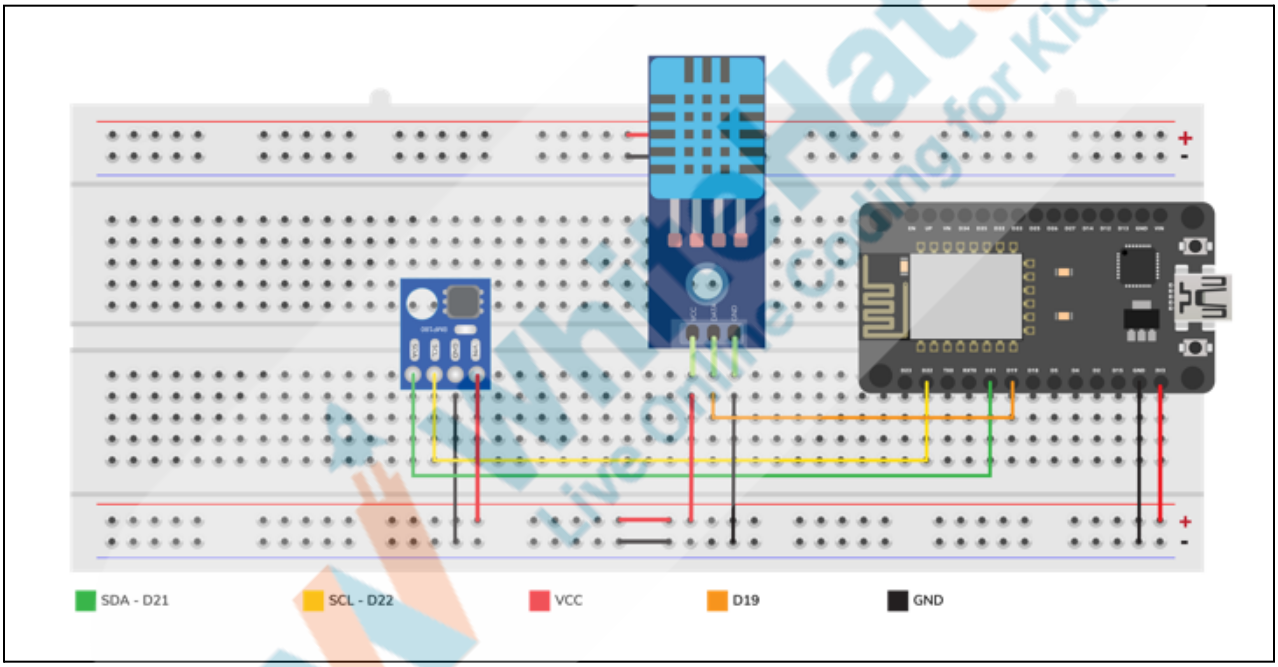
• The teacher gets into Full Screen.
Student Initiates Screen Share
ACTIVITY • **Breadboard Set Up** • **Arduino Program**

Teacher Action	Student Action
We have set up our firestore database. The next task is to get the values from the **ESP32** board.	
Step -1: Gather the following devices from your 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



Include libraries

The top of the program has the includes. We will 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 requests.

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);
```

After uploading libraries the next step is to connect with ESP32 with the WiFi. For that, we need to 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"
```

Initialize the setup()

- **Serial. begin(9600)** is used for data exchange speed. This tells the Arduino to get ready to exchange messages with the Serial Monitor at a

data rate of 9600 bits per second. That's 9600 binary ones or zeros per second and is commonly called a baud rate.

- **Serial. println** is used to print the statement
- This section will show details while connecting to the Internet. Basically, it will show when it gets connected with Wi-FI
- **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();
}
```

To execute the main process write the **void loop()**

- **Serial.print is** used to print data
- **readTemperature()** will read the temperature of the surroundings and print the result in **celsius.**

- **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);
}
```

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

Note: If the port is not selected, insert the USB cable in Computer's port and select the port

```
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.

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

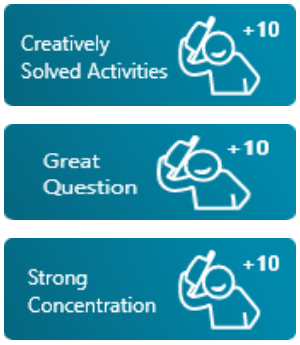
Activity Details

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 servers and how to feed value in the database.	*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

ACTIVITY LINKS		
Activity Name	**Description**	**Links**
Teacher Activity 1	Google Firebase	https://firebase.google.com/
Teacher Activity 2	Reference Code –Sensors	https://github.com/procodingclass/PRO-C251-Reference-Code
Teacher Reference 1	IN-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/10eea66d-3bcd-481e-b8bb-c362195a4e23.docx
Student Activity 1	Google Firebase	https://firebase.google.com/