CS G518: IoT

First Semester 2023-24

Assignment -1

Topic: Developing an IoT Sensor Data API and Data Collection Client

Instructor in charge - Dr. Ashutosh Bhatia

SUBMITTED BY

JOSHI KASHYAP - 2022H1030116P

API Endpoints and Their Functionality:

1) /temperature (GET):
This API endpoint retrieves real-time temperature data from a DHT11 temperature and humidity sensor connected to the Raspberry Pi. The data is returned in JSON format, encapsulated within the key "temperature."

```
h20220119@raspberrypi:~ $ curl http://172.18.19.135:3000/temperature
{"temperature":29.0}
```

2) /humidity (GET):
The /humidity endpoint serves to obtain current humidity data from the same DHT11 sensor. Similar to temperature, the data is provided in JSON format, labeled under the key "humidity."

```
h20220119@raspberrypi:~ $ curl http://172.18.19.135:3000/humidity
{"humidity":40.0}
```

3) /sound (GET):
The /sound API endpoint is responsible for fetching the present sound level recorded by a sound sensor attached to the Raspberry Pi. The result is conveyed as an integer, representing the sound level.

```
h20220119@raspberrypi:~ $ curl http://172.18.19.135:3000/sound
{"sound_level":1}
```

4) /fetch-all (GET):
The /fetch-all endpoint combines the functionalities of /temperature, /humidity, and /sound. It consolidates these sensor readings into a single JSON object, encompassing keys such as "temperature," "humidity," and "sound."

```
h20220119@raspberrypi:~ $ curl http://172.18.19.135:3000/fetch_all
{"humidity":40.0,"sound_level":1,"temperature":29.0}
```

Client Script Functionality(client.py):

- The client script, client.py, serves as an interface for users to interact with the Raspberry Pi-based server. Here's a step-by-step explanation of its operation:

- Initialization: The script initiates a connection to the Raspberry Pi server, identified by the address 172.18.19.135 and port 3000.

- User Menu: Users are presented with a menu that offers various options:

  "1": Retrieve and display current temperature data.
  "2": Fetch and display current humidity data.
  "3": Acquire and display current sound level data.
  "4": Fetch all sensor data (temperature, humidity, and sound) in a single request.
  "q": Quit the program.

Option Selection: Depending on the chosen option, the script sends an HTTP GET request to the corresponding API endpoint on the Raspberry Pi server.

- **Response Handling**: The client script processes the server's response as follows:

- For sensor data (temperature, humidity, or sound), the script displays the data on the console and inserts it into a local SQLite database.
- In the event of an error or an invalid option, an error message is presented.
- Repeat Action: The script waits for 2 seconds before repeating the chosen action (excluding "q") in a loop, ensuring periodic data updates.

```
h20220119@raspberrypi:~ $ python3 client.py
Select From Below Option
1:Temperature
2:Humidity
3.Sound
4.Fetch All Sensor Data
q:Quit
1
Temperature: 29.00
Data inserted into database: temperature: 29.0
```

```
h20220119@raspberrypi:~ $ python3 client.py
Select From Below Option
1:Temperature
2:Humidity
3.Sound
4.Fetch All Sensor Data
q:Quit
4
Humidity: 40.0
Data inserted into database: humidity: 40.0
Sound_level: 1
Data inserted into database: sound_level: 1
Temperature: 29.0
Data inserted into database: temperature: 29.0
```

Local Database Schema: -

- Created A Local Database named **“dbSensor.db”**
-Table Name :- sensor_data
-Note:- No need to run database setup scripts separately; the client.py script will automatically create all the necessary database tables when executed.

Column Name	Data type	Constraint	Comment
id	INTEGER	PRIMARY KEY AUTOINCREMENT	
sensor_type	TEXT		This will store type of sensor : temperature,humidity,sound
value	REAL		
timestamp	DATETIME	DEFAULT CURRENT_TIMESTAMP	

Output:-