Efficient Computing for Artificial Intelligence
Homework 3

DUE DATE: 31 Jan (h23:59)

Submission Instructions:

Each group will send an e-mail to `daniele.jahier@polito.it` and `valentino.peluso@polito.it` (in cc) with subject **ECAI26 Group**`N` (replace `N` with the group ID).

Attach a single ZIP archive named **HW3_Group**`N`**.zip** (replace `N` with your group ID) containing:

1. The code deliverables specified in the text of each exercise.
2. A one-page PDF report named **report.pdf**.

Late messages or messages not compliant with the above specifications will be **automatically discarded**.

Smart Hygrometer – Version 3.0

Goal: Design and develop data communication, retrieval, and visualization components of the Smart Hygrometer system using standard IoT and cloud communication protocols, specifically **MQTT** and **REST**.

System Description

The system is composed of the following components:

- A **Raspberry Pi** (RPI) connected to a **DHT-11 sensor** for measuring temperature and humidity.
- A **Redis Cloud database** used to store sensor data, accessed through the **redis-py API**.
- A **Deepnote MQTT subscriber** that receives temperature and humidity data published by the RPI and store received data to the Redis Cloud database.
- A **Deepnote REST API** that allows retrieval of historical data from the Redis Cloud database.
- A **Deepnote visualization dashboard** that retrieves historical data through the REST API and displays it using plots.

1. Data Collection, Communication, and Storage

1.1. Data Collection and Communication with the MQTT Protocol

On the RPI, develop a Python script named `publisher.py` that performs the following operations every 5 seconds:

a) Measure temperature and humidity using the **DHT-11** sensor.
b) Create a JSON message containing the measured data using the format described below.
c) Publish the JSON message to a public MQTT broker.

Message Format

The JSON message must include the fields described in Table 1.

Parameter	Description
mac_address	(string) MAC address of the RPI network interface, used to uniquely identify the device.
timestamp	(integer) Time of measurement expressed as Unix timestamp in milliseconds.
data	(list) List of sensor measurements. Each element contains: • `name`: (string) name of the measured quantity (`temperature` or `humidity`) • `value`: (integer) measured value

Table 1: Structure of the MQTT JSON message.

An example message is shown below:

```
{
  "mac_address": "0xf0b61e0bfe09",
  "timestamp": 1664630309530,
  "data": [
    {"name": "temperature", "value": 20},
    {"name": "humidity", "value": 65}
  ]
}
```

MQTT Communication

The script must publish the JSON message using the **MQTT protocol** with the following settings:

- **Broker address:** `broker.emqx.io`
- **Port:** 1883
- **Topic:** student ID of one member of the group (e.g., `s001122`)

1.2. Data Storage

On Deepnote, create a new Python notebook named `subscriber.ipynb` to implement an MQTT subscriber. The subscriber must receive the temperature and humidity messages published by the RPI and store the received data in the Redis Cloud database.

For data storage, use two **Redis TimeSeries**:

- `<mac_address>:temperature`
- `<mac_address>:humidity`

where `<mac_address>` is the MAC address of the RPI acting as MQTT publisher (e.g., `0xf0b61e0bfe09:temperature` and `0xf0b61e0bfe09:humidity`).

Each received MQTT message must be parsed to:

- Extract the timestamp.
- Extract temperature and humidity values.
- Append the values to the corresponding Redis TimeSeries using the extracted timestamp.

Run the `subscriber.ipynb` notebook while data is being published by the RPI using the script developed in Section 1.1.

2. Data Management & Visualization

2.1. REST Server

On Deepnote, duplicate the notebook `9.9 - Lab5 - Sensor Analytics REST API` from the Material project and extend the existing REST API to allow users to retrieve historical temperature and humidity data stored in Redis. The API must be extended by introducing a new resource and endpoint, described below.

Resource: `HistoricalData`

The `HistoricalData` resource represents a set of historical measurements collected by a device.

Parameter	Description
`mac_address`	(string) MAC address of the Raspberry Pi network interface.
`timestamp`	(list of integers) Each value is a timestamp expressed in milliseconds.
`temperature`	(list of integers). Each value represents a temperature measurement in °C.
`humidity`	(list of integers) Each value represents a humidity measurement in %RH.

Table 2: Fields of the `HistoricalData` resource.

Resource Example

```
{
  "mac_address": "0xf0b61e0bfe09",
  "timestamp": [1664630309530, 1664630310567, 1664630311542],
  "temperature": [22, 22, 23],
  "humidity": [60, 61, 59]
}
```

Endpoint /data/{mac_address}

HTTP Method: ____ /data/{mac_address} (fill the blank space with the most appropriate HTTP method)

Description: Retrieve the most recent *count* temperature and humidity samples collected by the device identified by the specified MAC address.

Hint: Use the Redis TimeSeries command `revrange()` to retrieve the most recent samples.

Path Parameters

Parameter	Description
`mac_address`	String (required). MAC address of the Raspberry Pi device.

Query Parameters

Parameter	Description
`count`	Integer (required). Number of most recent temperature and humidity samples to retrieve. Must be > 0.

Request Body

Not applicable.

Response Status Codes

- **200 – OK:** Request processed successfully.
- **400 – Bad Request:** Missing MAC address.
- **400 – Bad Request:** Missing count parameter.
- **400 – Bad Request:** The value of count is not a positive integer.
- **404 – Not Found:** MAC address not found in the database.

Response Schema

`HistoricalData`

Response Example

```
{
  "mac_address": "0xf0b61e0bfe09",
  "timestamp": [1664630309530, 1664630310567, 1664630311542],
  "temperature": [22, 22, 23],
  "humidity": [60, 61, 59]
}
```

2.2. REST Client

On Deepnote, create a Python notebook named `rest_client.ipynb` to implement a REST client and a visualization dashboard. The notebook should sequentially perform the following actions:

a) Check the server status using the endpoint `GET /status`.
b) Add your RPI sensor node using `POST /sensors`.
c) Retrieve the 10 most recent temperature and humidity samples collected by your RPI using the endpoint implemented in Section 2.1.
d) Plot the retrieved data using Deepnote chart blocks:
   i) Line plot of **Timestamp** (x-axis) vs. **Temperature** (y-axis).
   ii) Line plot of **Timestamp** (x-axis) vs. **Humidity** (y-axis).

3. Report

Prepare a one-page PDF report to answer the discussion questions reported below:

1. Explain why MQTT is a better choice than REST as the communication protocol for transmitting data from the RPI to the cloud (Exercise 1)

2. Specify the most suitable HTTP method (GET, POST, PUT, or DELETE) that has been used for implementing the required functionalities for the /data/mac_address endpoint and motivate your answer (Exercise 2).

4. Evaluation

Each homework part contributes separately to the final grade as reported in Table 3. This homework must be submitted only by students enrolled in **Data Science and Engineering (DSE)**.

Exercise Part	**DSE (max points)**
1. Data Collection, Communication, and Storage	2
2. Data Management & Visualization	2
3. Report	2
Total	**6**

Table 3: Evaluation scheme for DSE students.

5. Deliverables

Submit a single ZIP archive named: **HW3_GroupN.zip** containing:

1. `publisher.py` – Python script that contains the code of Section 1.1. The code is intended to be run on the RPI without installing additional software or dependencies on the provided SD card. Moreover, the script must include all necessary classes and methods needed for its correct execution.
2. `subscriber.ipynb` – Python notebook that contains the code of Section 1.2. The notebook is intended to be run on Deepnote, must use only the packages that get installed with the *ECAI Template* and must run without any additional dependency.
3. `rest_server.ipynb` – Python notebook that contains the code of Section 2.1. The notebook is intended to be run on Deepnote, must use only the packages that get installed with the *ECAI Template* and must run without any additional dependency.
4. `rest_client.ipynb` – Python notebook that contains the code of Section 2.2. The notebook is intended to be run on Deepnote, must use only the packages that get installed with the *ECAI Template* and must run without any additional dependency.
5. `report.pdf` – One-page report.

Note: Submissions that do not follow the required filenames or format **will be discarded without evaluation**.