Authored just now by Pruthviraj Subudhi

Python Assignment API for Intern Hiring 2021 Assignment

General Instruction

Mandatorily required from candidates 2 Links

1. Public link for the application hosted
2. Public repository link of your source code.

You need to write a flask/Django application, to do the following things and your API should be live on And APIs should be publicly accessible to us to verify. (Ex. It can be deployed in AWS, GCP or Heroku, Pythonanywhere)

Note:

- Consider all the dates and times are in the UTC timezone, if not mentioned.
- Please create unique API points for each question.

Question 1

- URL to use: https://gitlab.com/-/snippets/2094509/raw/master/sample_json_1.json
- Use the given URL to get the sample_json_1. The given sample JSON represents the production state of two units of a plant.
- Design a Flask/Django API which takes in start time and end time as a query string and returns the number of times a production unit state is in running state for the asked duration.
- The plant functions in three different shifts as given below (timings are in IST timezone):

> shiftA : 6:00 AM - 2:00 PM
>
> shiftB : 2:00 PM - 8:00 PM
>
> shiftC : 8:00 PM - 6:00 AM

Points to Note:

1. The resultant JSON should show the count shift-wise.
2. start time and end time query string format: "%Y-%m-%dT%H:%M:%SZ"
3. Value True represents the running state, False represents the idle state
4. expected JSON format

```
{
        "shiftA" :{ "production_A_count" :50, "production_B_count" :25},
        "shiftB" :{ "production_A_count" :50, "production_B_count" :25},
        "shiftC" :{ "production_A_count" :50, "production_B_count" :25},
}
```

5. eg input -

```
start_time = "2021-01-28T07:30:00Z"
end_time = "2021-01-28T13:30:00Z"
```

eg output -

```
{
        "shiftA" :{ "production_A_count" :2, "production_B_count" :3},
        "shiftB" :{ "production_A_count" :9, "production_B_count" :7},
        "shiftC" :{ "production_A_count" :0, "production_B_count" :0},
}
```

Question 2

- URL to use: https://gitlab.com/-/snippets/2094509/raw/master/sample_json_2.json
- Use the given URL to get the sample_json_2. The given sample JSON gives the recorded runtime and downtime of a machine at a given time.
- Design a Flask/Django API which takes in start time and end time as a query string and returns the total runtime, downtime, and machine utilization in the asked interval.

Points to Note:

1. start time and end time query string format: "%Y-%m-%dT%H:%M:%SZ"
2. runtime and downtime are in secs.
3. max runtime can be 1021 secs, if value greater than this consider the exceding mins as downtime.
4. machine utilisation = (total runtime)/(total runtime + total downtime) * 100
5. expected JSON format:

```
{
        "runtime" : "2h:30m:46s",
        "downtime": "1h:20m:29s",
        "utilisation": 64.63
}
```

6. eg input -

```
start_time = "2021-01-28T08:30:00Z"
end_time = "2021-01-28T10:30:00Z"
```

output -

```
{
        "runtime" : "1h:53m:17s",
        "downtime": "0h:49m:06s",
        "utilisation": 69.76
}
```

Question 3

- URL to use: https://gitlab.com/-/snippets/2094509/raw/master/sample_json_3.json
- Use the given URL to get the sample_json_3. Design a Flask/Django API which takes in start time and end time as a query string and returns the average value of belt1 and belt2 for unique id's in the given time interval.

Points to Note:

1. start time and end time query string format: "%Y-%m-%dT%H:%M:%SZ"
2. - belt1 value is to be considered 0 when the state is True and
   - belt2 value is to be considered 0 when the state is False.
3. remove the string part of the id and change the datatype of "id" to integer in returned JSON
4. set the ids in ascending order
5. expected JSON format:

```
[
        {"id" : 2, "avg_belt1" : 2200, "avg_belt2" :3300},
        {"id" : 4, "avg_belt1" : 2200, "avg_belt2" :3300},
        {"id" : 7, "avg_belt1" : 2200, "avg_belt2" :3300}
]
```

6. eg input -

```
start_time = "2021-01-28T18:30:00Z"
end_time = "2021-01-28T20:10:00Z"
```

output -

```
[
        {"id" : 4, "avg_belt1" : 1198, "avg_belt2" : 0},
        {"id" : 7, "avg_belt1" : 0, "avg_belt2" : 1316},
        {"id" : 8, "avg_belt1" : 1431, "avg_belt2" : 0},
        {"id" : 9, "avg_belt1" : 1918, "avg_belt2" : 0},
]
```

7. Output Explantion:

```
start_time = "2021-01-28T18:30:00Z"
end_time = "2021-01-28T20:10:00Z"
```

Records for id ch004 are

```
{
        "time": "2021-01-28 18:50:00",
        "id": "ch004",
        "state": false,
        "belt1": 1174,
        "belt2": 1278
},{
        "time": "2021-01-28 20:10:00",
        "id": "ch004",
        "state": false,
        "belt1": 1222,
        "belt2": 1692
}
```

> avg_belt1 = (1174 + 1222)/2 = 1198

> avg_belt2 = 0

1.py 7 bytes

```
print()
```