Question1:

Purpose: Reports is most important data end users would look for which may be anything from analytics report, stock market analytics report, invoice, ticket booking etc.. Everything happens automated in backend based on user input.
Backend server would generate the pdf and stream pdf to frontend to download or trigger a mail to end user.

Task: Create an automated pdf report using python which has graph capabilities. The final pdf report must contain the graphs (bar chart, line graph, pie chart), table data and some paragraph data. The input to the pdf would be json. DOwnload the input data from https://drive.google.com/file/d/1di7sjYHFRa_0cx4--P1ytQizWgSw-hGp/view?usp=share_link

The below image shows partial json with keys app_graph, time_graph, alerts, and kl_data.


```
{
   "app_graph": [{
       "app": "Linux Terminal",
       "time": "2h 40m",
       "percent": "53.9"
   },
   ....
   ],
   "time_graph": [{
       "name": "Active key Time",
       "time": 27
   }, ....],
   "alerts": [{
       "event_id": 441,
       "rule": 1,
       "alert_id": "08f9873d1c1c2c64548b11b8a14ab578a3df33295149b5d0fd5e653fbb5e2063",
       "sha1": "1795aee923935d4ea59e08f9a6abe21d46ba9a24",
       "match": {
           "end": 16,
           "match": "14.15.67.89",
           "start": 5,
           "substr": false,
           "match_type": "ip",
```

```
            "compute_time": 0
        },
        "time": "2022-05-26 21:48:08",
        "level": 2,
        "is_array": true,
        "rule_name": "Sensitive Data Detection in Keylogging Data",
        "tag": ["DLP", "PII"]
    }, ....],
    "kl_data": "\nAgain wronng,\n\nu"
}
```


Expected Output:

1. Put Header and footer for pdf. Header would be some random Name lets say DUMMY PDF Check. Footer would be your Iit Jummu address, and page number of pdf.
2. It should have **bar graph** representing app_graph data, where **app_graph.app** would be X-axis and Y-axis is Percent.
3. Immediately beside the **bar graph,** you should get a **Table** with COls: S.No, Name (**app_graph.app**), Time (**app_graph.time**).
4. From 2 and 3, It looks like the bar graph is placed on the left side and the right side is the table. Each covering half of the width of the pdf page.
5. Should have **pie chart**, access time_graph, where time_graph.time would be the pie chart sector value and the legend for pie chart would be time_graph.name
6. Should contain a full width **table**, access **alerts** key. Table cols S.No(event_id), Alert Id(alert_id), sha1 (sha1), Time (time), Name (rule_name), Data Match (match.match)
7. And finally end your pdf with **paragraph** by accessing **kl_data** key
8. **Compute the time** taken to generate the pdf report and share the information. Check the cpu utilization when pdf is generated.
9. Do **benchmarking**, run the pdf generation and output to same filename, run it 100 times, 1000 times and calculate the time taken to run 100 times, 1000 times, and operations/second or in other words, in 1 second how many pdfs are generated. At the same time while benchmarking, track the cpu consumption during entire course of run. Share the results of benchmarking in .md file and upload to github. Create separate file bench.py to run bench marking code which imports your python script and calls pdf generation function.

Resources:

You can refer to:

https://plotly.com/python/v3/pdf-reports/ (recommended)

https://towardsdatascience.com/how-to-create-a-pdf-report-for-your-data-analysis-in-python-2bea81133b
https://towardsdatascience.com/how-to-create-pdf-reports-with-python-the-essential-guide-c08dd3ebf2ee

Sharing Output:

- Pdf generation code
- Benchmarking code bench.js
- .md file containing benchmarking reports
- Sample pdf output.

Share above files via Github, create a private repo in github from your account, and add **mai1x9** as **collaborator.**