TASK 2
Creating a server

In this task you'll be expected to create your own local HTTP server that hosts raw data contained in a text (.txt) file

The file would contain < 100 lines, each containing a random BITS ID of various batches and branches in the usual format Example: 2024B4PS0950P etc.

You're expected to read the data from the file and display it on various endpoints specified below.

But before that

API Parameters
API parameters are inputs sent with an API request that define and/or modify the data you'll be interacting with

This task involves 2 of them which are directly included in the URL of the API namely

1. path parameters
2. query parameters

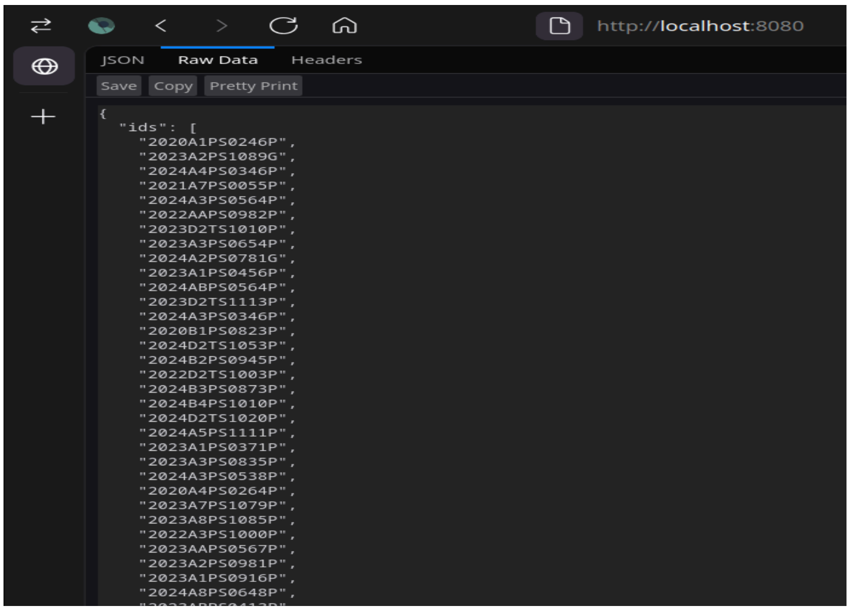
path parameters are usually represented in theory as ***{url}/:param*** (don't be confused, when making a request to this url you will NOT include the colon) where param is the path parameter which could be anything, and the API responds to the request based on this parameter. These are generally used to identify specific resources or subset of an API, usually referring to a unique resource such as a product, user etc

query parameters are represented as ***{url}?param=value*** where param is the actual query parameter and value is the value of that parameter, which again could be anything and the API responds to the request based on the value of this parameter. These are generally used for a restricted number of values and usually for filtering, sorting, paginating data accessed by specifying a path parameter.

Now back to the task.

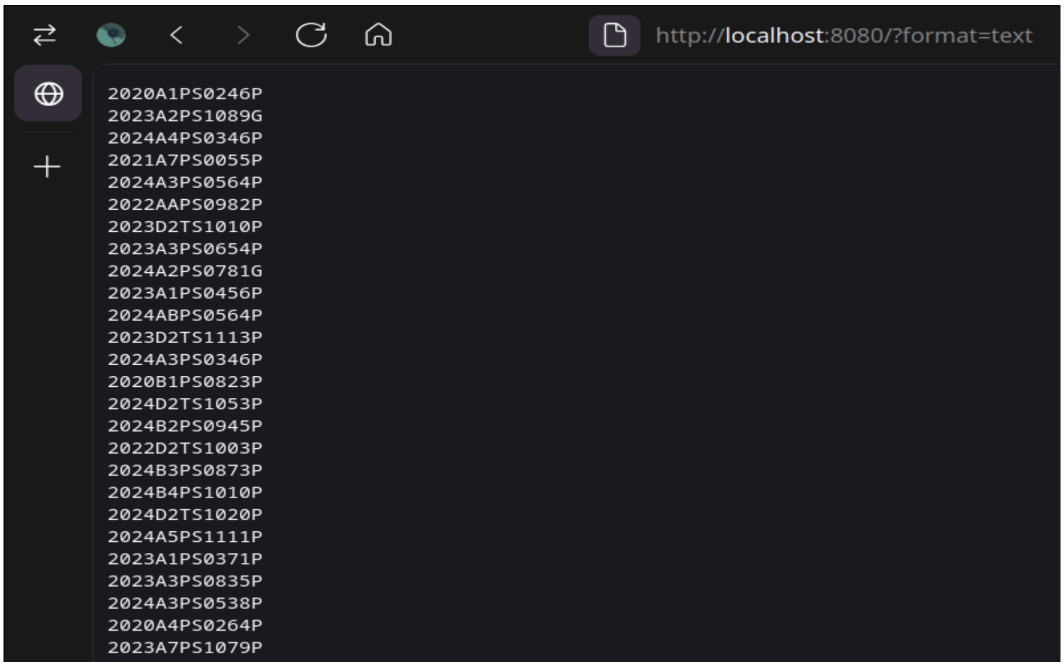
Let's say the server is hosted on *localhost:8000*
and let's call this ***{{base_url}}*** from now on

- ***{{base_url}}***
  responds with all the ID's in the JSON format containing the key "ids" and the value being a list of all the ID's as in the file



```
{
  "ids": [
    "2020A1PS0246P",
    "2023A2PS1089G",
    "2024A4PS0346P",
    "2021A7PS0055P",
    "2024A3PS0564P",
    "2022AAPS0982P",
    "2023D2TS1010P",
    "2023A3PS0654P",
    "2024A2PS0781G",
    "2023A1PS0456P",
    "2024ABPS0564P",
    "2023D2TS1113P",
    "2024A3PS0346P",
    "2020B1PS0823P",
    "2024D2TS1053P",
    "2024B2PS0945P",
    "2022D2TS1003P",
    "2024B3PS0873P",
    "2024B4PS1010P",
    "2024D2TS1020P",
    "2024A5PS1111P",
    "2023A1PS0371P",
    "2023A3PS0835P",
    "2024A3PS0538P",
    "2020A4PS0264P",
    "2023A7PS1079P",
    "2023A8PS1085P",
    "2022A3PS1000P",
    "2023AAPS0567P",
    "2023A2PS0981P",
    "2023A1PS0916P",
    "2024A8PS0648P",
```

- **{{base_url}}**?format=text
  responds with all the ID's in plain text format the same way as in the file

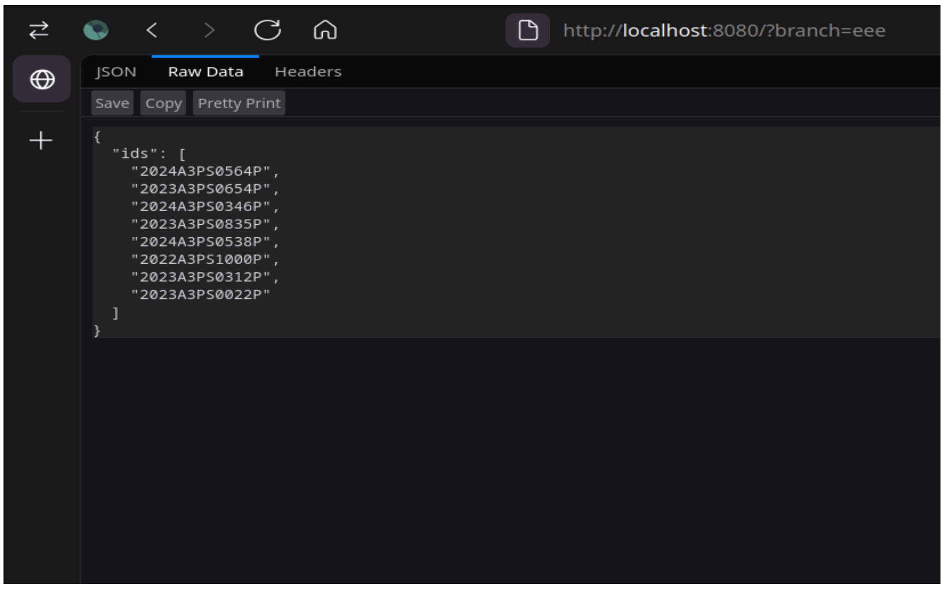


```
2020A1PS0246P
2023A2PS1089G
2024A4PS0346P
2021A7PS0055P
2024A3PS0564P
2022AAPS0982P
2023D2TS1010P
2023A3PS0654P
2024A2PS0781G
2023A1PS0456P
2024ABPS0564P
2023D2TS1113P
2024A3PS0346P
2020B1PS0823P
2024D2TS1053P
2024B2PS0945P
2022D2TS1003P
2024B3PS0873P
2024B4PS1010P
2024D2TS1020P
2024A5PS1111P
2023A1PS0371P
2023A3PS0835P
2024A3PS0538P
2020A4PS0264P
2023A7PS1079P
```

- ***{{base_url}}***?branch=X
  where X can be anything of the following
  1. cs
  2. ece
  3. eee
  4. eni
  5. mech
  6. civil
  7. phy

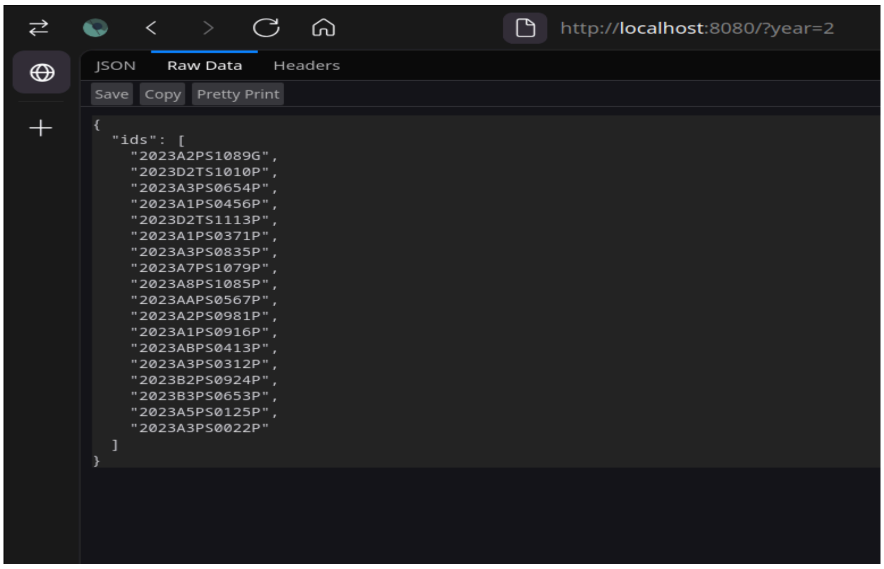
8. chem
9. chemical
10. math
11. bio
12. eco
13. pharma
14. Manu
15. genstudies

Which filters ID’s based on their branches and responds with a JSON object as in the previous endpoint



```
{
  "ids": [
    "2024A3PS0564P",
    "2023A3PS0654P",
    "2024A3PS0346P",
    "2023A3PS0835P",
    "2024A3PS0538P",
    "2022A3PS1000P",
    "2023A3PS0312P",
    "2023A3PS0022P"
  ]
}
```

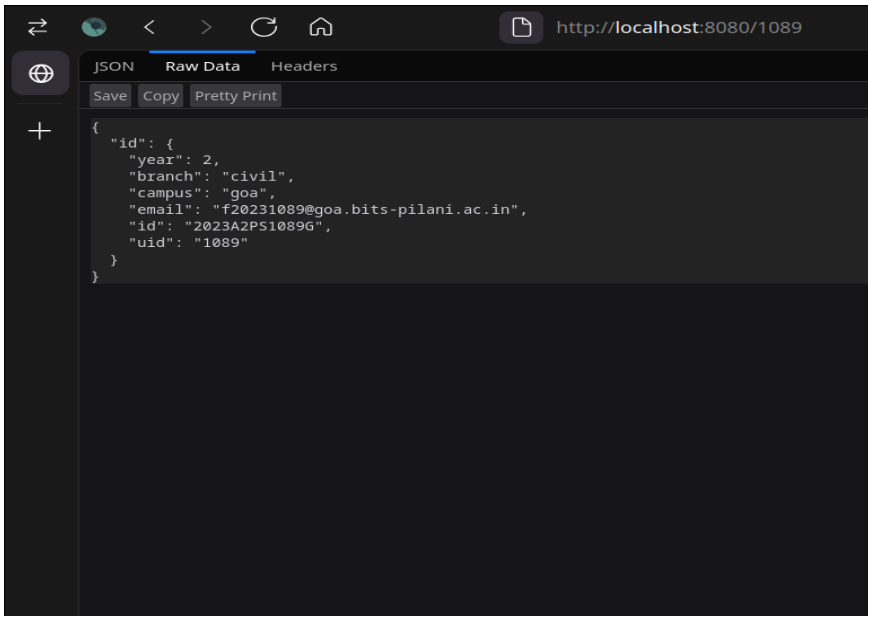
- ***{{base_url}}***?year=X
  where year represents the year they would be in right now (2024 - 1, 2023 - 2 and so on)



```
{
  "ids": [
    "2023A2PS1089G",
    "2023D2TS1010P",
    "2023A3PS0654P",
    "2023A1PS0456P",
    "2023D2TS1113P",
    "2023A1PS0371P",
    "2023A3PS0835P",
    "2023A7PS1079P",
    "2023A8PS1085P",
    "2023AAPS0567P",
    "2023A2PS0981P",
    "2023A1PS0916P",
    "2023ABPS0413P",
    "2023A3PS0312P",
    "2023B2PS0924P",
    "2023B3PS0653P",
    "2023A5PS0125P",
    "2023A3PS0022P"
  ]
}
```

- ***{{base_url}}***/:id

responds with a JSON object containing the keys “id”, “uid”, “email”, “branch”, “year”, “campus”



```
{
  "id": {
    "year": 2,
    "branch": "civil",
    "campus": "goa",
    "email": "f20231089@goa.bits-pilani.ac.in",
    "id": "2023A2PS1089G",
    "uid": "1089"
  }
}
```

NOTE:

- The program should take the path of the text file as the first argument and throw an error in any other case
- You need not implement cases for multiple query parameters, in such a case respond with a valid response for any one of the query parameter
- If there are two ID’s with the same four last digits, display info for only the first ID in the file
- In any case where there was no data to be found respond with a simple JSON object

  {“error”: “<What really happened>”}

  and an appropriate HTTP status code
- You can go to *localhost:port* where *port* is the port number your server is running on in a browser to check your code or use tools like ***curl*** if you're on linux or ***Invoke-WebRequest*** If you’re on Windows… Another great tool is ***HTTPie*** which is cross-platform and really user friendly.

You're free to use any of the following languages Go, Nodejs, Python (Go is recommended)

You're also required to have a basic understanding of git, GitHub as the solution to this needs to be uploaded to GitHub

Further things to try (not graded, but will be fun)

- There is really a lot of scope for additions here, handling duplicate unique identifiers (the four unique digits of your Id which might clash with people from different years), implementing cases where multiple query parameters could be provided, adding other query parameters like campus etc

- an interesting feature would be to add a ***-r*** flag which would take a directory and recurse into it and parse all the .txt files it encounters, and now, that recursion has come into picture adding a ***-L*** flag with the level to recurse into would also be something to consider.
- Try to find ways to optimise the solution if there are let’s say 1000s of lines in the text file

A Google Form will be shared later for collecting your responses where you will be submitting the GitHub repository link of your work