

PENTONIX

1. Given a string s, find the length of the longest substring without repeating characters.

Input: s = "abcabcbb"

Output: 3

Program:

```
#include <bits/stdc++.h>

using namespace std;

int findRes(string& inp) {
    unordered_map<char,int> map;
    int cur = 0, res = 0;
    for(int i = 0; i < inp.length(); i++) {
        if(map[inp[i]] == 0) {
            cur++;
            map[inp[i]] = 1;
        }
        else {
            res = max(res, cur);
            cur = 1;
            map.clear();
            map[inp[i]] = 1;
        }
    }
    res = max(res, cur);
    return res;
}

int main() {
    string inp;
    cin >> inp;
    cout << findRes(inp);
    return 0;
}
```

Explanation:

A map is being used to keep track of any repetition of characters in the string while iterating through the string.

2. To run the flask server, follow the below steps:

Clone the repository:

```
git clone https://github.com/S-Sanjith/pentonix-assignment.git  
cd pentonix-assignment
```

Install dependencies:

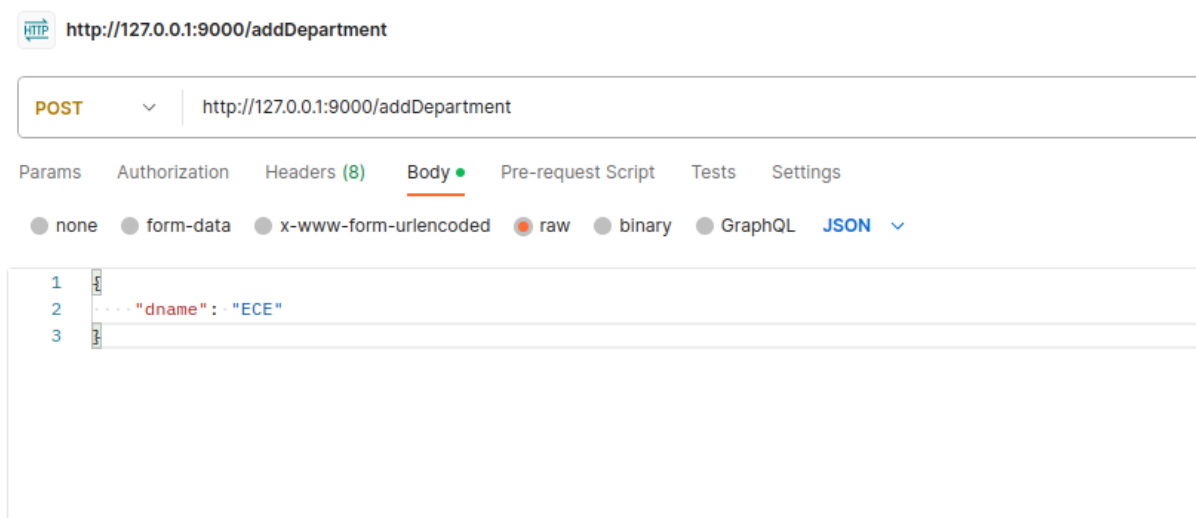
```
pip install -r requirements.txt
```

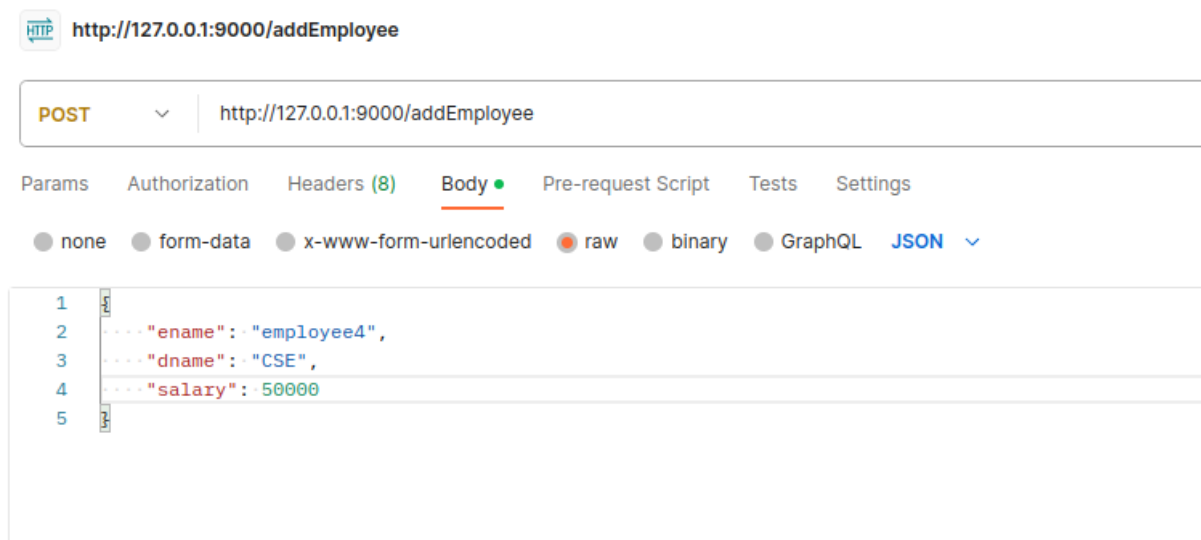
Run the Flask Server:

```
python server.py
```

The database consists of two tables Department and Employee.

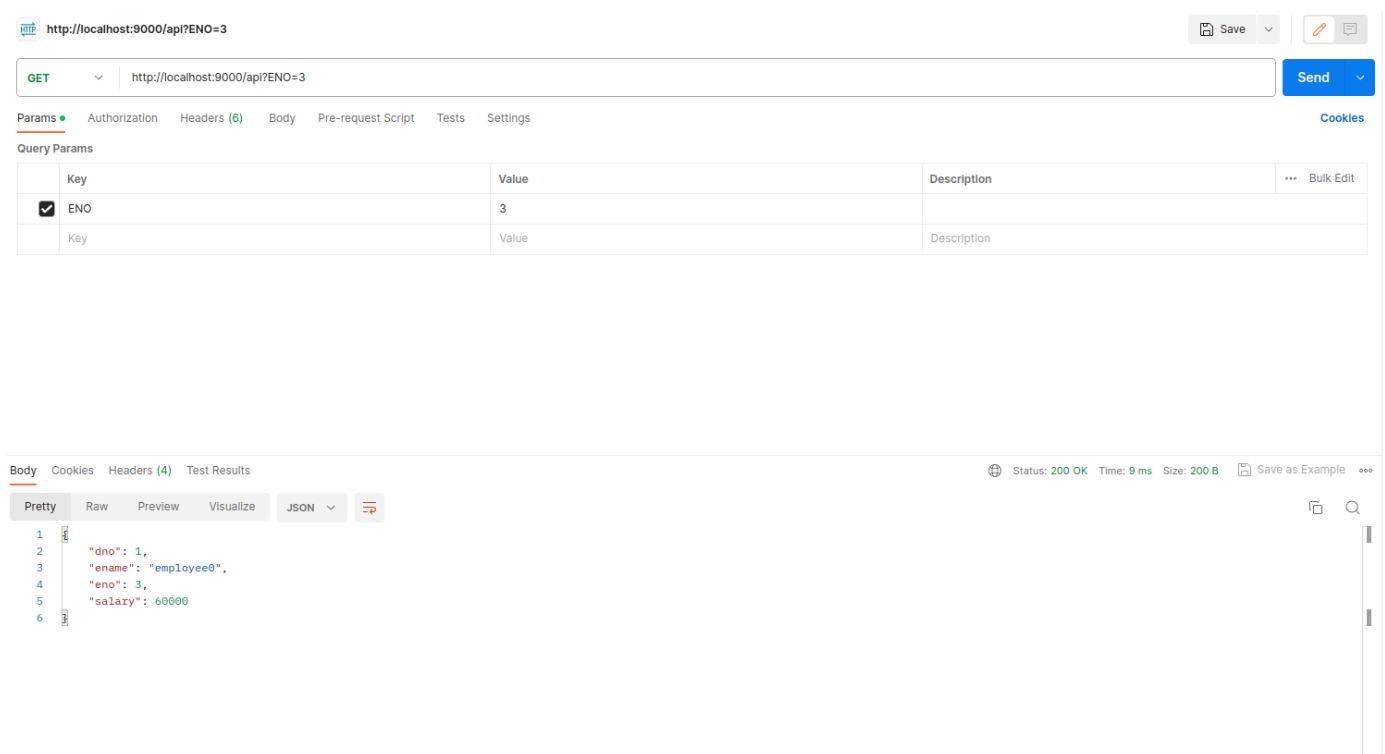
To insert data to the tables, use the routes `/addEmployee` and `addDepartment` via POST requests.





- a. To get details of employees when ENO is given as parameter:

`http://localhost:9000/api?ENO=1`



This gives the Employee details of employee with ENO=1 as response when we mention ENO=1 is mentioned as parameter in the request.

- b. To get the list of details of all the employees whose DNAME is mentioned as parameter in the request:

<http://localhost:9000/api?DNAME=CSE>

The screenshot displays a REST client interface. At the top, the URL bar shows `http://localhost:9000/api?DNAME=CSE`. Below it, the method is set to `GET` and the same URL is entered in the request field. The 'Query Params' section shows a table with one parameter:

Key	Value	Description
<input checked="" type="checkbox"/> DNAME	CSE	


The 'Body' tab is selected, showing the response in 'Pretty' JSON format. The status is `200 OK`, the time is `12 ms`, and the size is `256 B`. The JSON response is as follows:

```
1 {
2   {
3     "dno": 2,
4     "ename": "employee6",
5     "eno": 1,
6     "salary": 60000
7   },
8   {
9     "dno": 2,
10    "ename": "employee8",
11    "eno": 2,
12    "salary": 60000
13  }
14 }
```

This gives details of all Employees whose DNAME matches with the parameter.

To view all details of Employee and Department tables, use the `/allEmployees` and `/allDepartments` routes.

Employee:

 http://127.0.0.1:9000/allEmployees

GET

▼

http://127.0.0.1:9000/allEmployees

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Query Params

	Key	Value
	Key	Value

Body

Cookies

Headers (4)

Test Results


Pretty

Raw

Preview


Visualize

JSON ▼



```
1  {
2    "Employee": [
3      {
4        "dno": 2,
5        "ename": "employee6",
6        "eno": 1,
7        "salary": 60000
8      },
9      {
10       "dno": 2,
11       "ename": "employee8",
12       "eno": 2,
13       "salary": 60000
14     },
15     {
16       "dno": 1,
17       "ename": "employee0",
18       "eno": 3,
19       "salary": 60000
20     }
21   ]
22 }
```

Department:

 http://127.0.0.1:9000/allDepartments

GET

⌵

http://127.0.0.1:9000/allDepartments

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Query Params

	Key	Value
	Key	Value

Body

Cookies

Headers (4)

Test Results


Pretty

Raw

Preview

Visualize

JSON ⌵



1

2

3

4

5

6

7

8

9

10

11

12

⌵

"Department": [

{

"dname": "ISE",

"dno": 1

,

{

"dname": "CSE",

"dno": 2

]

⌵