Creating APIs in Flask

app.py

```
from flask import Flask
from flask_cors import CORS, cross_origin
```

config.py

```
from app import app
from flaskext.mysql import MySQL

mysql = MySQL()
app.config['MYSQL_DATABASE_USER'] = 'root'
app.config['MYSQL_DATABASE_PASSWORD'] = ''
app.config['MYSQL_DATABASE_DB'] = 'test'
app.config['MYSQL_DATABASE_HOST'] = 'localhost'
mysql.init_app(app)
```

main.py

```
app = Flask(__name__)
CORS(app)
import pymysql
from app import app
from config import mysql
from flask import jsonify
from flask import flash, request
```

```
@app.route('/create', methods=['POST'])
def create_emp():
    try:
        _json = request.json
        _name = _json['name']
        _email = _json['email']
        _phone = _json['phone']
        _address = _json['address']
        if _name and _email and _phone and _address and request.method == 'POST':
            conn = mysql.connect()
            cursor = conn.cursor(pymysql.cursors.DictCursor)
            sqlQuery = "INSERT INTO emp(name, email, phone, address)
VALUES(%s, %s, %s, %s)"
            bindData = (_name, _email, _phone, _address)
            cursor.execute(sqlQuery, bindData)
            conn.commit()
            respone = jsonify('Employee added successfully!')
            respone.status_code = 200
            return respone
        else:
            return showMessage()
    except Exception as e:
        print(e)
    finally:
        cursor.close()
        conn.close()
```

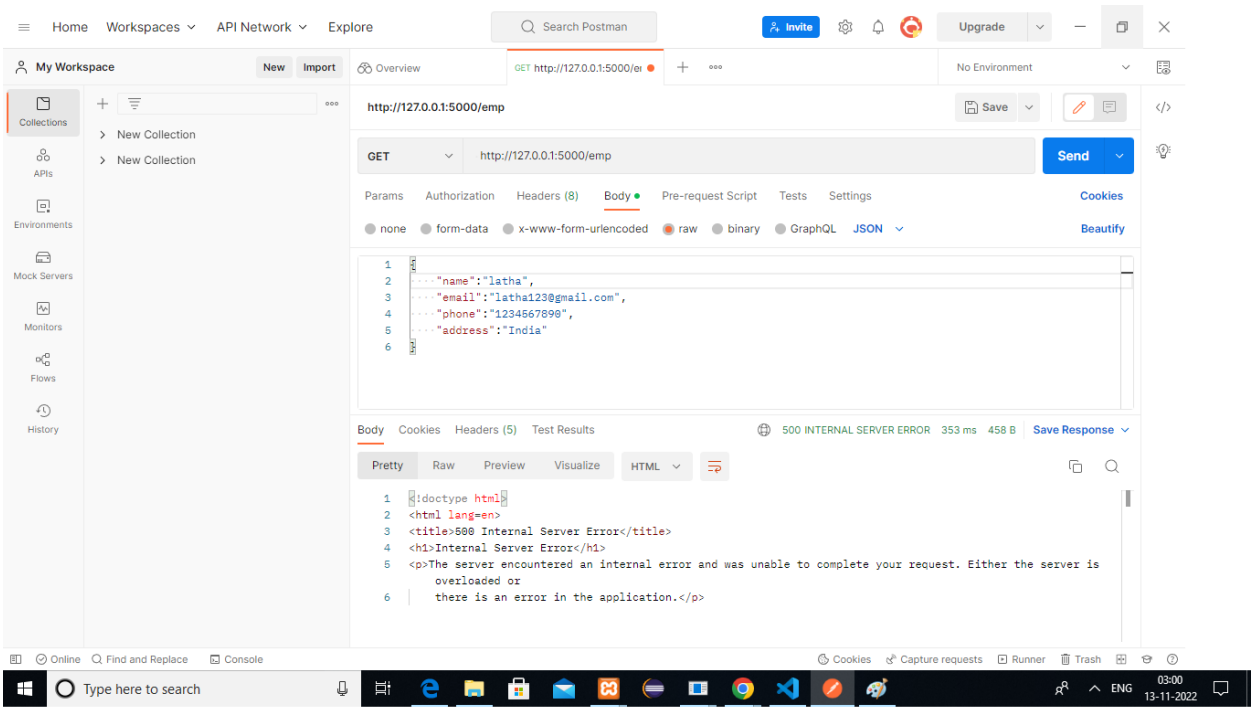
```
@app.route('/emp')
def emp():
    try:
        conn = mysql.connect()
        cursor = conn.cursor(pymysql.cursors.DictCursor)
        cursor.execute("SELECT id, name, email, phone, address FROM emp")
        empRows = cursor.fetchall()
        respone = jsonify(empRows)
        respone.status_code = 200
        return respone
    except Exception as e:
        print(e)
    finally:
        cursor.close()
        conn.close()


@app.route('/emp/')
def emp_details(emp_id):
    try:
        conn = mysql.connect()
        cursor = conn.cursor(pymysql.cursors.DictCursor)
        cursor.execute("SELECT id, name, email, phone, address FROM emp
WHERE id =%s", emp_id)
        empRow = cursor.fetchone()
        respone = jsonify(empRow)
```

```
        respone.status_code = 200
        return respone
    except Exception as e:
        print(e)
    finally:
        cursor.close()
        conn.close()


@app.route('/update', methods=['PUT'])
def update_emp():
    try:
        _json = request.json
        _id = _json['id']
        _name = _json['name']
        _email = _json['email']
        _phone = _json['phone']
        _address = _json['address']
        if _name and _email and _phone and _address and _id and request.method ==
'PUT':
            sqlQuery = "UPDATE emp SET name=%s, email=%s, phone=%s,
address=%s WHERE id=%s"
            bindData = (_name, _email, _phone, _address, _id,)
            conn = mysql.connect()
            cursor = conn.cursor()
            cursor.execute(sqlQuery, bindData)
            conn.commit()
```

```
            respone = jsonify('Employee updated successfully!')
            respone.status_code = 200
            return respone
        else:
            return showMessage()
    except Exception as e:
        print(e)
    finally:
        cursor.close()
        conn.close()

@app.route('/delete/', methods=['DELETE'])
def delete_emp(id):
        try:
                conn = mysql.connect()
                cursor = conn.cursor()
                cursor.execute("DELETE FROM emp WHERE id =%s", (id,))
                conn.commit()
                respone = jsonify('Employee deleted successfully!')
                respone.status_code = 200
                return respone
        except Exception as e:
                print(e)
        finally:
                cursor.close()
```

```
                conn.close()


@app.errorhandler(404)
def showMessage(error=None):
    message = {
        'status': 404,
        'message': 'Record not found: ' + request.url,
    }
    respone = jsonify(message)
    respone.status_code = 404
    return respone

if __name__ == "__main__":
    app.run()
```

Output


Home
Workspaces
API Network
Explore
Search Postman
Invite
Upgrade
My Workspace
New
Import
Overview
GET http://127.0.0.1:5000/er
No Environment
Collections
APIs
Environments
Mock Servers
Monitors
Flows
History
New Collection
New Collection
http://127.0.0.1:5000/emp
Save
GET
http://127.0.0.1:5000/emp
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"name":"latha",
"email":"latha123@gmail.com",
"phone":"1234567890",
"address":"India"
}
Body
Cookies
Headers (5)
Test Results
500 INTERNAL SERVER ERROR
353 ms
458 B
Save Response
Pretty
Raw
Preview
Visualize
HTML
<!doctype html>
<html lang=en>
<title>500 Internal Server Error</title>
<h1>Internal Server Error</h1>
The server encountered an internal error and was unable to complete your request. Either the server is overloaded or
there is an error in the application.
Online
Find and Replace
Console
Cookies
Capture requests
Runner
Trash
Type here to search
ENG
03:00
13-11-2022