

# Docker Assignment

[Github link : <https://github.com/shubham8596/DockerAssignment> ]

Software/Plugins/Language Used :

- Python
- Flask Web Framework
- Postman
- Mysql
- Postman
- Docker

Implementation :

Step 1 :

I have done all installations.  
e.g. flask and its dependencies, postman

Step 2 :

Create the folder Structure in that one folder contain REST API(CRUD) operations and another folder contain mysql file. In first folder i have write all configuration and business logic for REST API. In second folder i have write sql file to create database and table.

Step 3 :

Write Dockerfile which contain all code and mysql.

Step 4 :

Build Dockerfile :

```
sudo docker build -t sample .
```

Run Dockerfile :

```
sudo docker run -it -p 8080:8080 sample
```

Step 5 :

In postman send the request -> POST,GET, PUT,DELETE

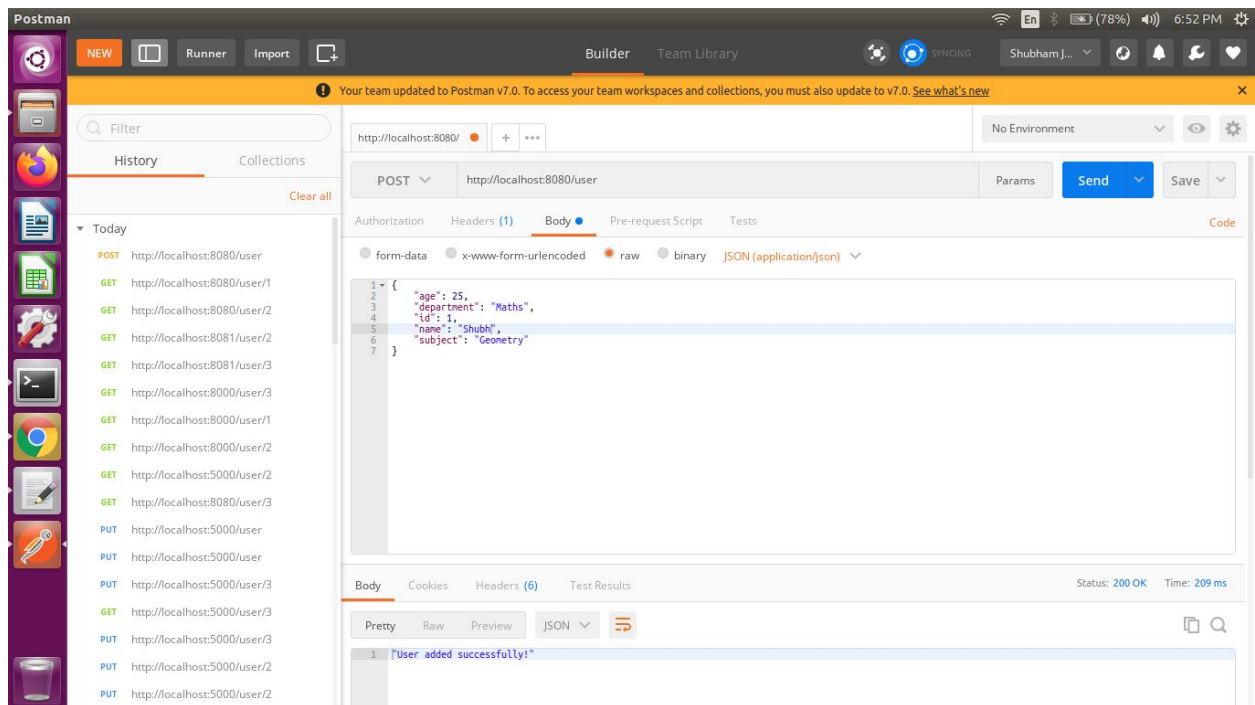
POST = Create data

GET = Read data

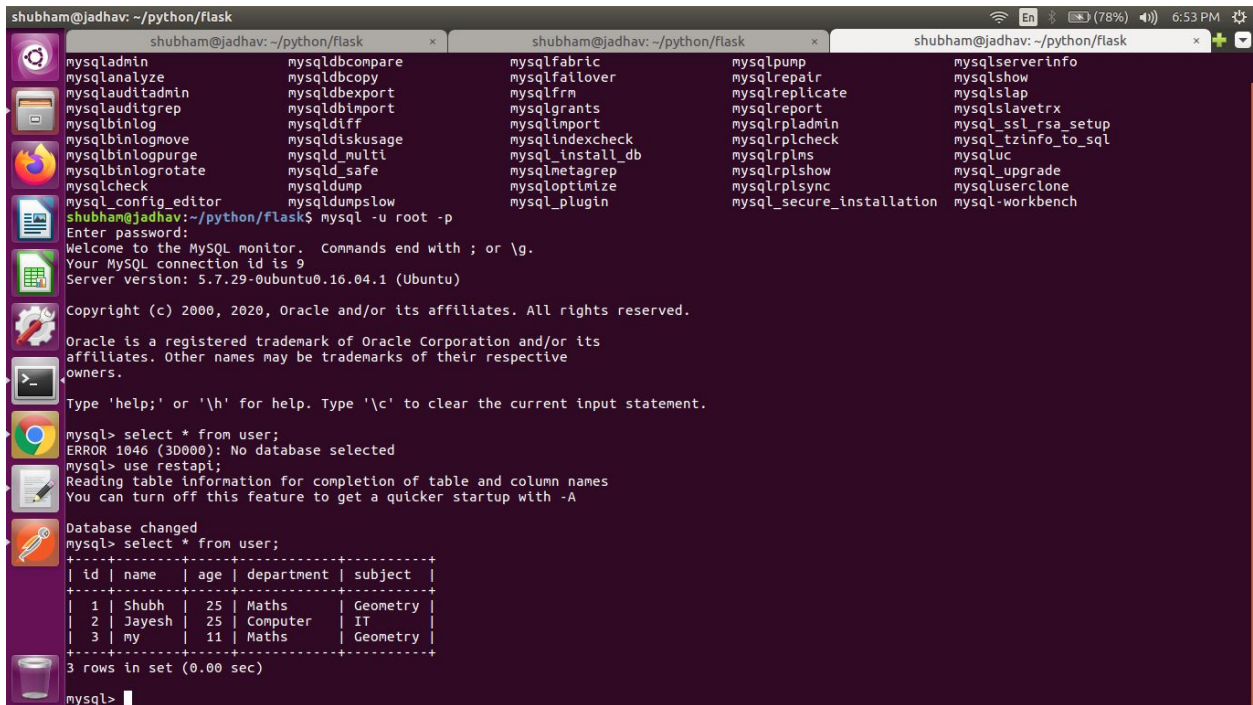
PUT = Update data

DELETE = Delete data

POST input :



## POST Output :

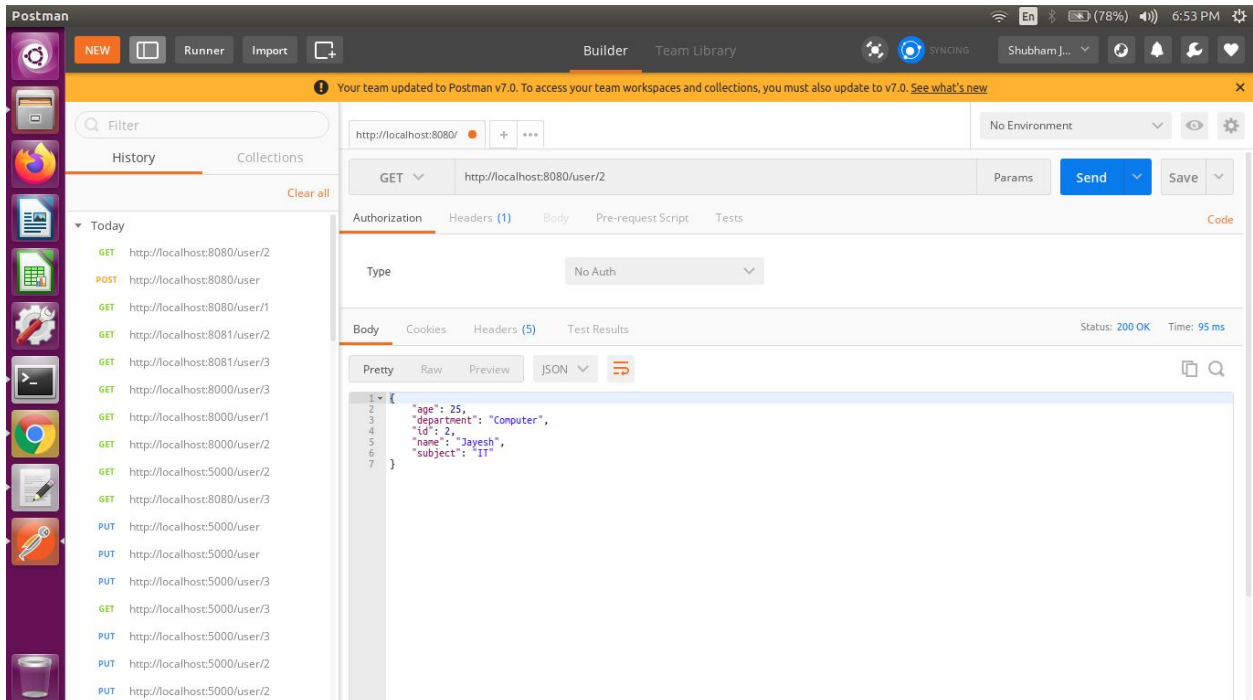


The terminal shows a MySQL command-line interface session. The user is logged in as root. The output displays the MySQL version (5.7.29-0ubuntu0.16.04.1) and the server's copyright information. The user then executes the command `mysql> select * from user;`, which results in an error: `ERROR 1046 (3D000): No database selected`. The user then executes `mysql> use restapi;`, which changes the database to `restapi`. Finally, the user executes `mysql> select * from user;`, which returns the following table:

id	name	age	department	subject
1	Shubh	25	Maths	Geometry
2	Jayesh	25	Computer	IT
3	my	11	Maths	Geometry

The output also indicates that 3 rows were returned in 0.00 seconds.

## GET Input :

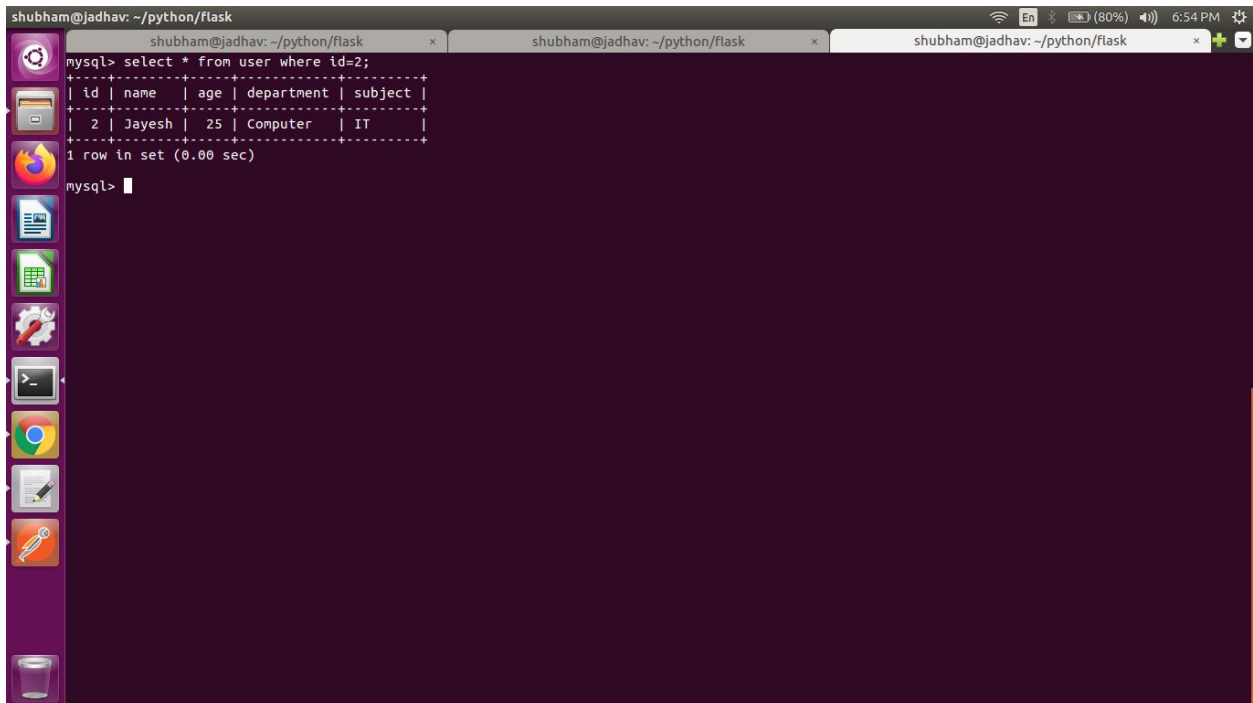


The screenshot shows the Postman application interface. A GET request is defined with the URL `http://localhost:8080/user/2`. The request is sent, and the response is displayed in the "Body" tab. The response is a JSON object with the following structure:

```
{  "age": 25,  "department": "Computer",  "id": 2,  "name": "Jayesh",  "subject": "IT"}
```

The status of the response is 200 OK, and the time taken is 95 ms.

## GET Output :

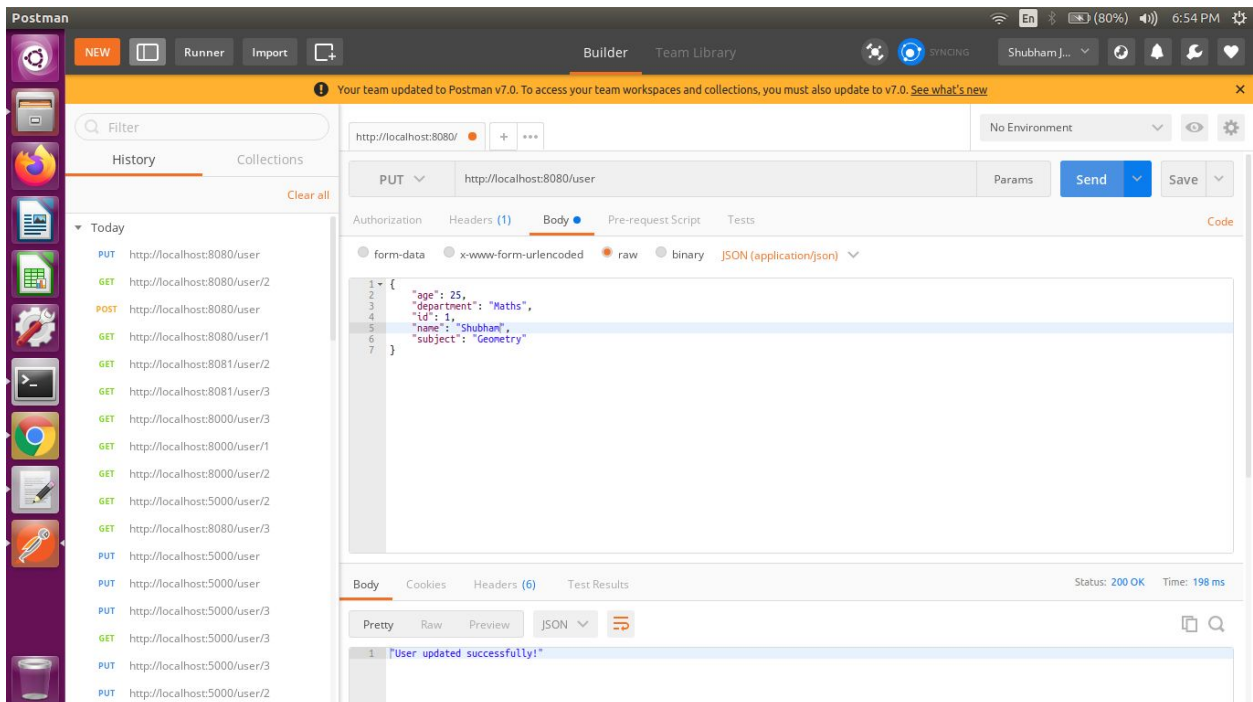


A terminal window showing a MySQL query execution. The query is `select * from user where id=2;`. The output is a single row with the following data:

id	name	age	department	subject
2	Jayesh	25	Computer	IT

The terminal also shows the message "1 row in set (0.00 sec)".

## PUT Input :

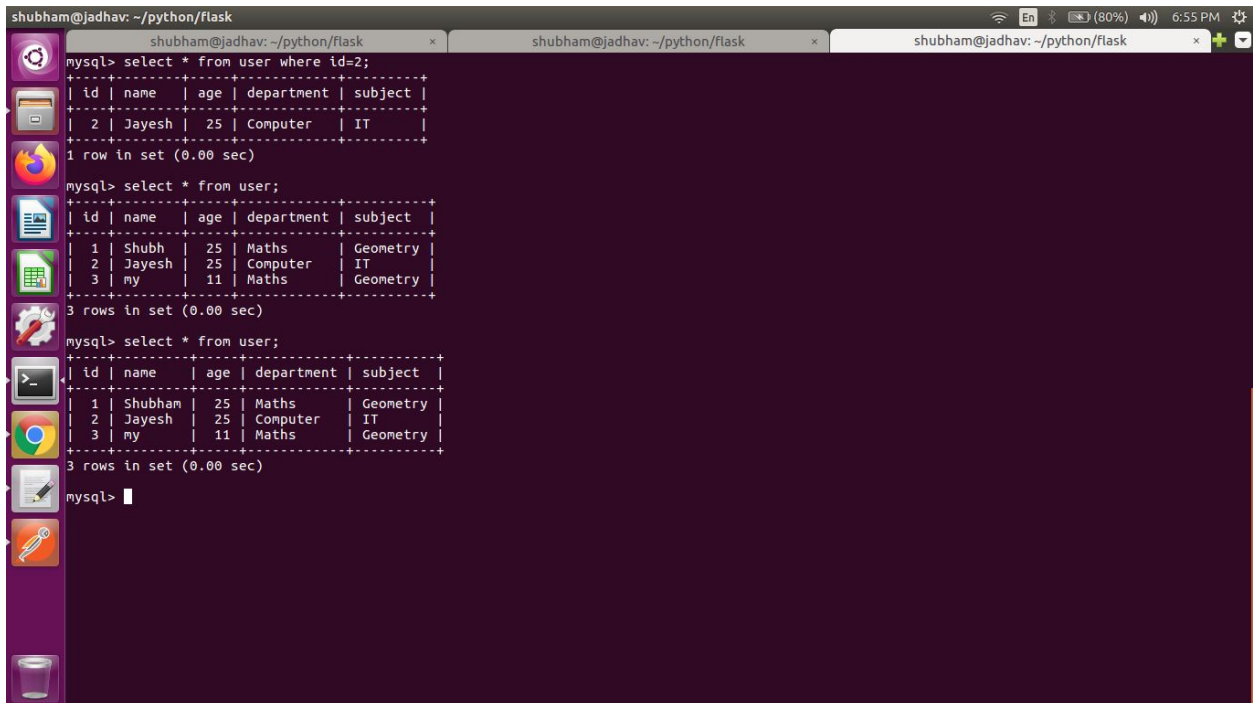


A screenshot of the Postman application showing a PUT request to `http://localhost:8080/user`. The request body is a JSON object:

```
{  "age": 25,  "department": "Maths",  "id": 1,  "name": "Shubham",  "subject": "Geometry"}
```

The response status is `200 OK` and the response body is `"User updated successfully!"`.

## PUT Output :



A terminal window showing MySQL commands and their output. The terminal has a dark purple background with light green text. The commands and output are as follows:

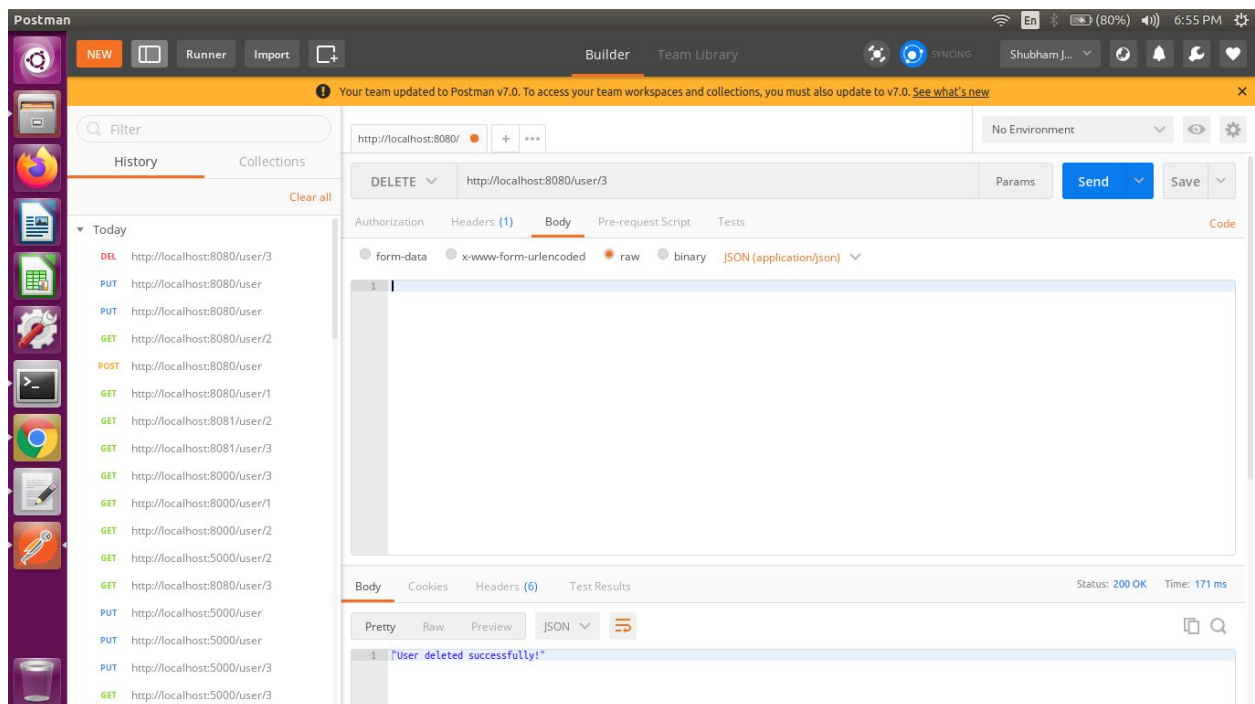
```
mysql> select * from user where id=2;
+----+-----+-----+-----+-----+
| id | name  | age  | department | subject |
+----+-----+-----+-----+-----+
| 2  | Jayesh | 25   | Computer  | IT       |
+----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> select * from user;
+----+-----+-----+-----+-----+
| id | name  | age  | department | subject |
+----+-----+-----+-----+-----+
| 1  | Shubh  | 25   | Maths      | Geometry |
| 2  | Jayesh | 25   | Computer  | IT       |
| 3  | my     | 11   | Maths      | Geometry |
+----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> select * from user;
+----+-----+-----+-----+-----+
| id | name  | age  | department | subject |
+----+-----+-----+-----+-----+
| 1  | Shubham | 25   | Maths      | Geometry |
| 2  | Jayesh  | 25   | Computer  | IT       |
| 3  | my      | 11   | Maths      | Geometry |
+----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

## DELETE Input :



## DELETE Output :

```
shubham@jadhav: ~/python/flask
mysql> select * from user where id=2;
+----+-----+-----+-----+-----+
| id | name  | age  | department | subject |
+----+-----+-----+-----+-----+
| 2  | Jayesh | 25   | Computer  | IT       |
+----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> select * from user;
+----+-----+-----+-----+-----+
| id | name  | age  | department | subject |
+----+-----+-----+-----+-----+
| 1  | Shubh  | 25   | Maths     | Geometry |
| 2  | Jayesh | 25   | Computer  | IT       |
| 3  | my     | 11   | Maths     | Geometry |
+----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> select * from user;
+----+-----+-----+-----+-----+
| id | name  | age  | department | subject |
+----+-----+-----+-----+-----+
| 1  | Shubham | 25   | Maths     | Geometry |
| 2  | Jayesh | 25   | Computer  | IT       |
| 3  | my     | 11   | Maths     | Geometry |
+----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> select * from user;
+----+-----+-----+-----+-----+
| id | name  | age  | department | subject |
+----+-----+-----+-----+-----+
| 1  | Shubham | 25   | Maths     | Geometry |
| 2  | Jayesh | 25   | Computer  | IT       |
+----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

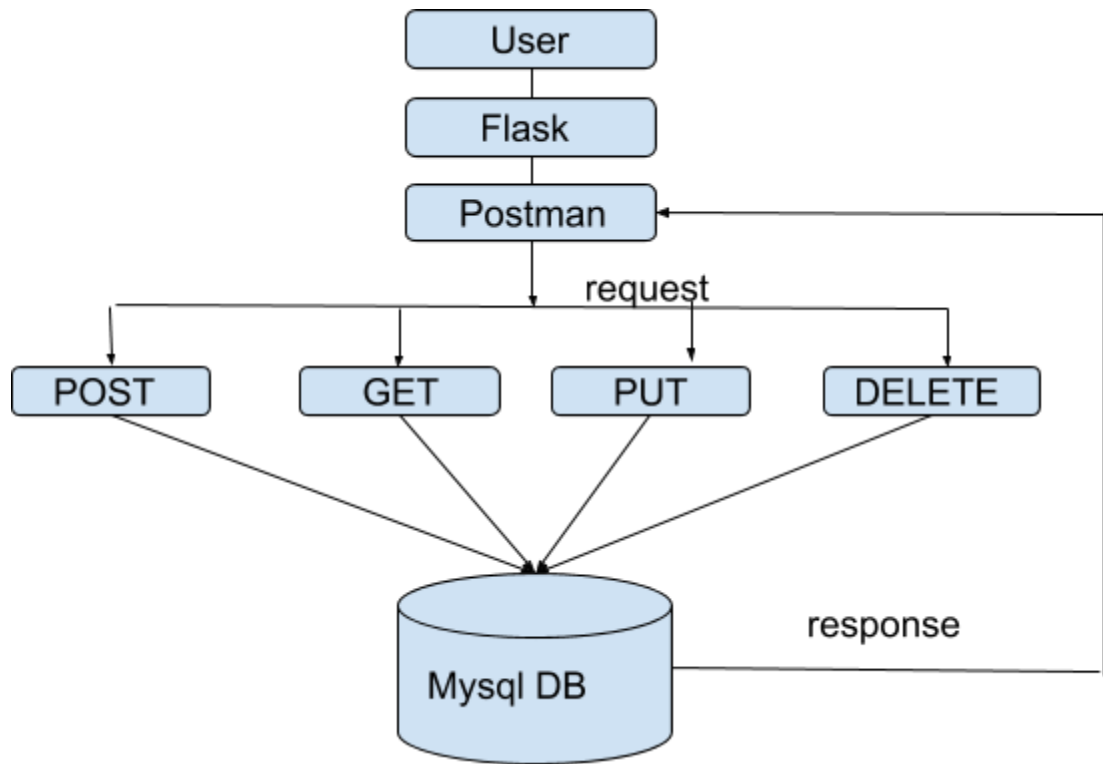
mysql>
```

## REST API :

```
shubham@jadhav: ~/python/flask
shubham@jadhav:~/python/flask$ python3 main.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:8080/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 130-007-500

127.0.0.1 - - [25/Apr/2020 18:51:16] "GET /user/2 HTTP/1.1" 200 -
127.0.0.1 - - [25/Apr/2020 18:51:22] "GET /user/1 HTTP/1.1" 200 -
127.0.0.1 - - [25/Apr/2020 18:51:46] "POST /user HTTP/1.1" 200 -
127.0.0.1 - - [25/Apr/2020 18:53:35] "GET /user/2 HTTP/1.1" 200 -
127.0.0.1 - - [25/Apr/2020 18:54:39] "PUT /user HTTP/1.1" 200 -
127.0.0.1 - - [25/Apr/2020 18:54:47] "PUT /user HTTP/1.1" 200 -
127.0.0.1 - - [25/Apr/2020 18:55:20] "DELETE /user/3 HTTP/1.1" 200 -
```

FlowDiagram :



\*\*\*\*\* Thanks \*\*\*\*\*