FOXMULA ASSIGNMENT

Creating a user in sqlite database and fetching the user based on id using rest APIs

Technologies used : Python, Flask and SQLite database. Flask is a lightweight framework for web applications using python.

Why Flask ?
Django is another heavier framework that functions well for web projects and APIs, while providing modularity and detail. However since the task at hand was a lightweight task involving a single user database and some basic APIs, flask can handle it easily.

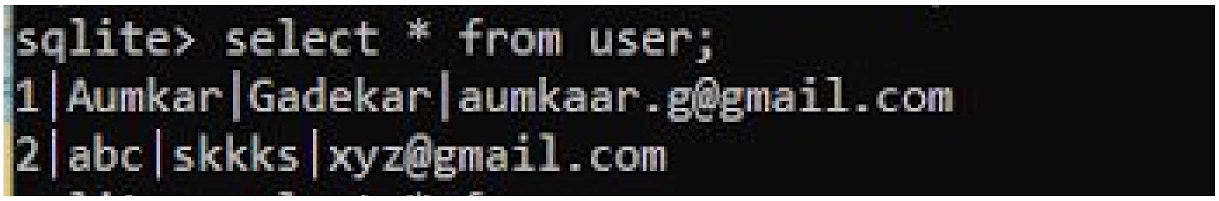
Code Walk-through :

The code for the apis is contained in the api.py file. The sqlite database is the users.db file. API testing was carried out using POSTMAN-API.

Importing libraries and setting up the app :

```
import flask
from flask import request, jsonify
from flask_restful import Resource, Api

import sqlite3

app = flask.Flask(__name__)
api=Api(app)
```

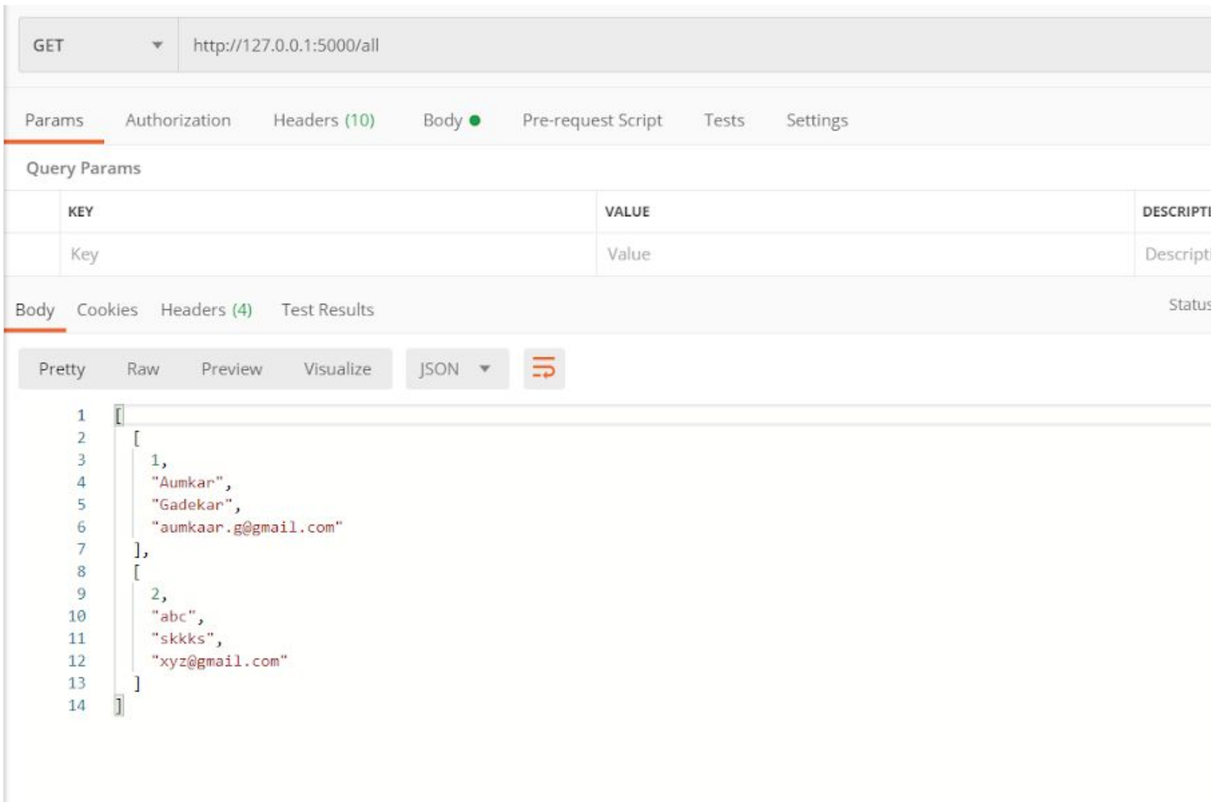
The database was created using SQLite and populated with 2 entries. Four columns - id, first name, last name and email describe the user.



```
sqlite> select * from user;
1|Aumkar|Gadekar|aumkaar.g@gmail.com
2|abc|skkks|xyz@gmail.com
```

GET API for displaying all users :

```
class all(Resource):
    def get(self):
        conn = sqlite3.connect('users.db')
        cur = conn.cursor()
        all_users= cur.execute('SELECT * FROM user;').fetchall()
        return jsonify(all_users)
```

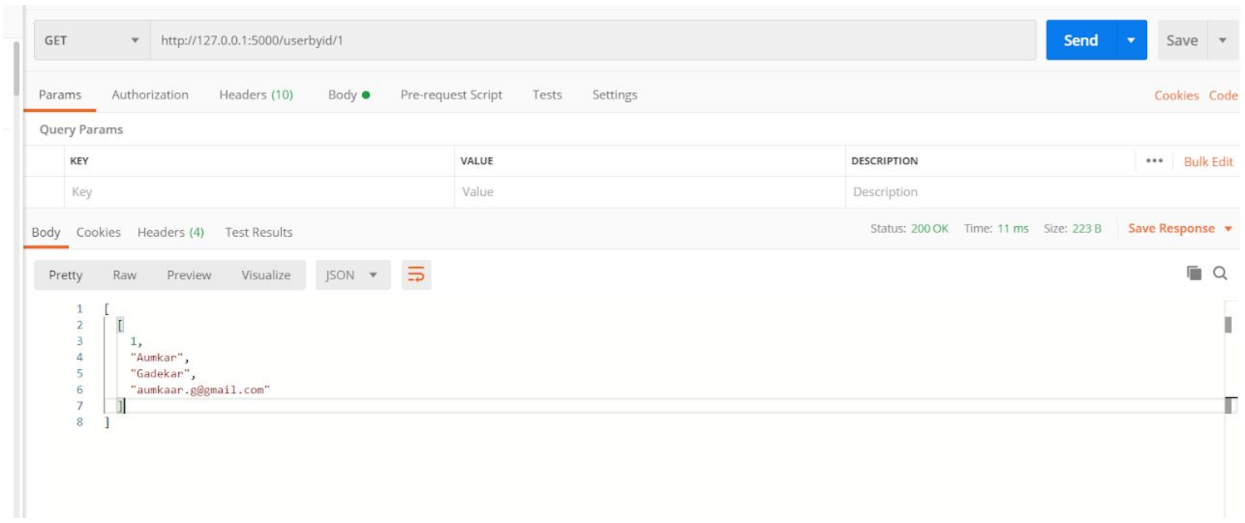


```
[
    [
        1,
        "Aumkar",
        "Gadekar",
        "aumkaar.g@gmail.com"
    ],
    [
        2,
        "abc",
        "skkks",
        "xyz@gmail.com"
    ]
]
```

Displays all users in the database.

GET API for displaying user based on id entered in parameter :

```python
class byid(Resource):
    def get(self,id):
        query = "SELECT * FROM user WHERE"
        to_filter = []
        if id:
            query += ' id=? AND'
            to_filter.append(id)
        if not (id):
            return page_not_found(404)

        query = query[:-4] + ';'
        conn = sqlite3.connect('users.db')
        cur = conn.cursor()
        results = cur.execute(query, to_filter).fetchall()
        return jsonify(results)
```



http://127.0.0.1:5000/userbyid/1

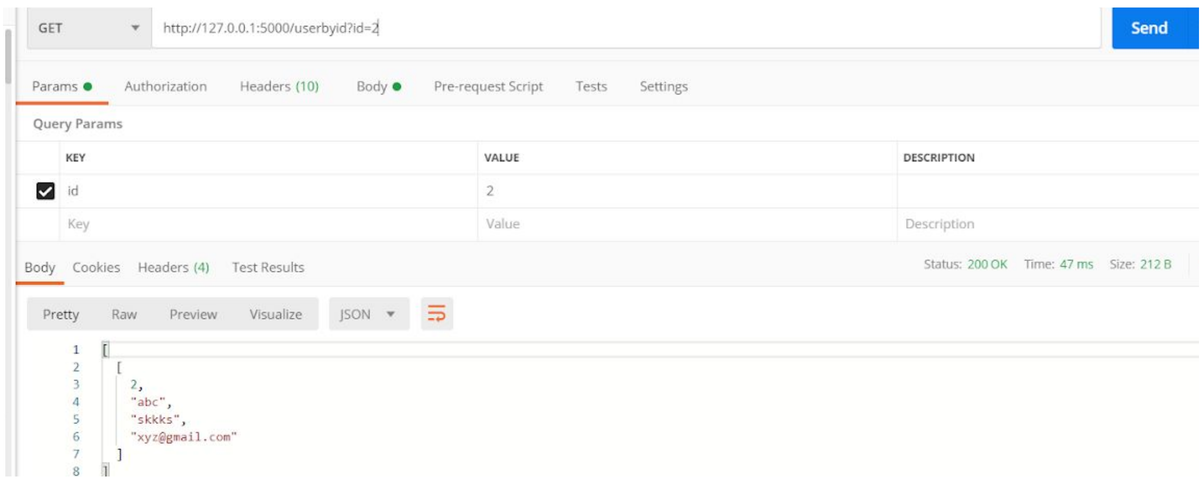
KEY	VALUE	DESCRIPTION
Key	Value	Description

Status: 200 OK Time: 11 ms Size: 223 B

```json
[
  [
    1,
    "Aumkar",
    "Gadekar",
    "aumkaar.g@gmail.com"
  ]
]
```

Get user by ID using query string :

```
@app.route('/userbyid', methods=['GET'])
def api_filter():
    query_parameters = request.args

    id = query_parameters.get('id')
    print(id)
    query = "SELECT * FROM user WHERE id=" +id+";"
    conn = sqlite3.connect('users.db')
    cur = conn.cursor()
    result = cur.execute(query).fetchall()
    if(len(result)==0):
        return jsonify({"error":"ID NOT FOUND"})
    return jsonify(result)
```



GET http://127.0.0.1:5000/userbyid?id=2

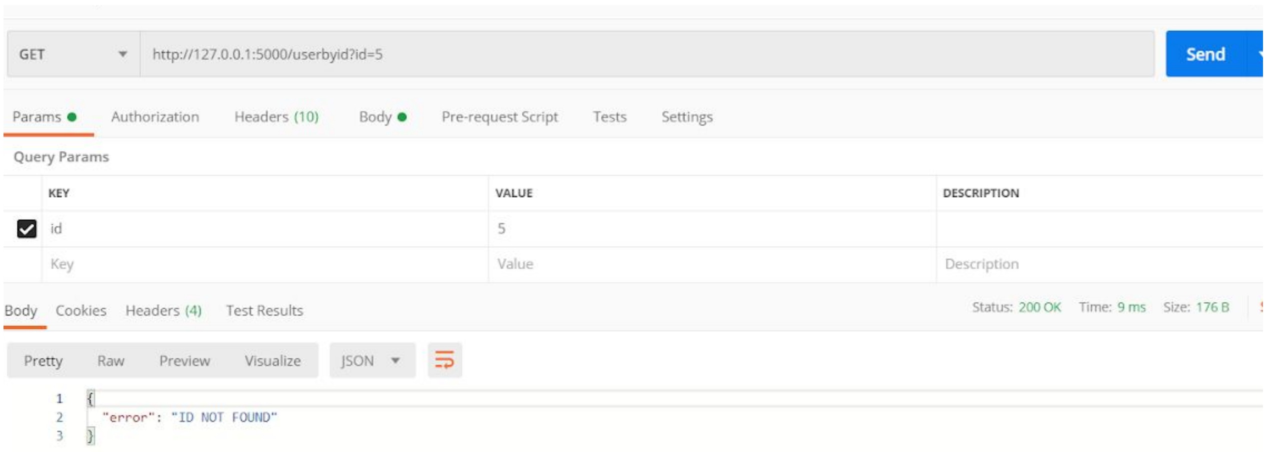
Query Params

KEY	VALUE	DESCRIPTION
id	2	

Status: 200 OK Time: 47 ms Size: 212 B

```
[
  [
    2,
    "abc",
    "skkks",
    "xyz@gmail.com"
  ]
]
```

Error message when ID doesn't exist :



GET http://127.0.0.1:5000/userbyid?id=5

Query Params

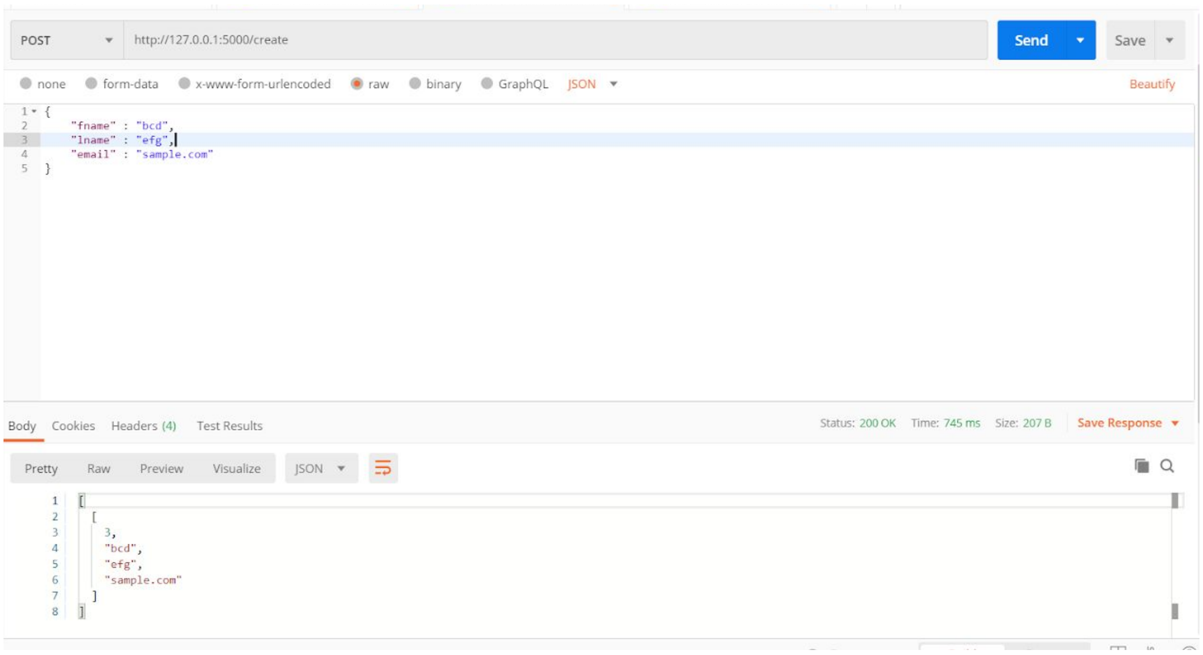
KEY	VALUE	DESCRIPTION
id	5	

Status: 200 OK Time: 9 ms Size: 176 B

```
{
  "error": "ID NOT FOUND"
}
```

Create new user with unique ID and return it :

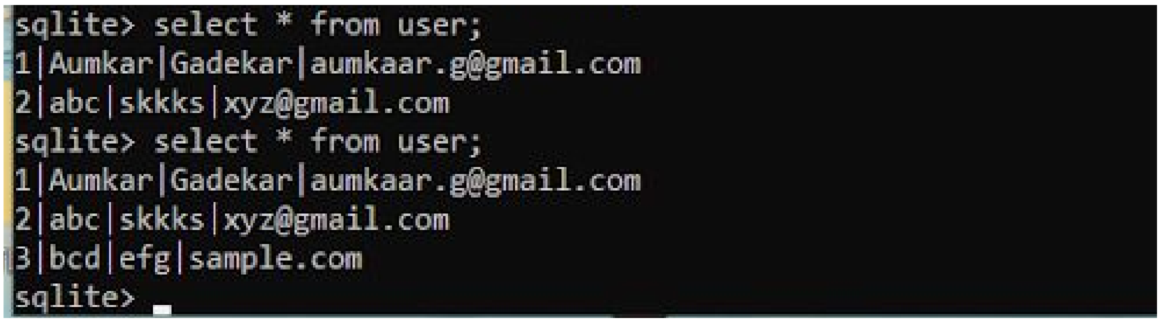
```
@app.route("/create", methods=['POST'])
def create():
    if request.method == 'POST':
        fname = request.json['fname']
        lname = request.json['lname']
        email = request.json['email']

        try:
            conn = sqlite3.connect('users.db')
            cur = conn.cursor()
            sql='SELECT * FROM user WHERE email = "'+email+'";'
            print(sql)
            email_match= cur.execute(sql).fetchall()
            if(email_match):
                return jsonify({'error':'email exists'})
            num=cur.execute("select num from info").fetchall()
            new_id=num[0][0]+1
            print(new_id)                                   #Generates
new ID
            sql="update info set num = "+str(new_id)+";"
            cur = conn.cursor()
            cur.execute(sql)
            conn.commit()
            cur.execute( "INSERT INTO user(id,fname,lname,email)
VALUES(? ,? ,? ,?);",(new_id,fname,lname,email))
            conn.commit()
            query = "SELECT * FROM user WHERE id=" +str(new_id)+";"
            result = cur.execute(query).fetchall()
            return jsonify(result)
        except Exception as e:
            print(e)
            return "Exception encountered"
        return page_not_found
```



```
{
    "fname" : "bcd",
    "lname" : "efg",
    "email" : "sample.com"
}
```

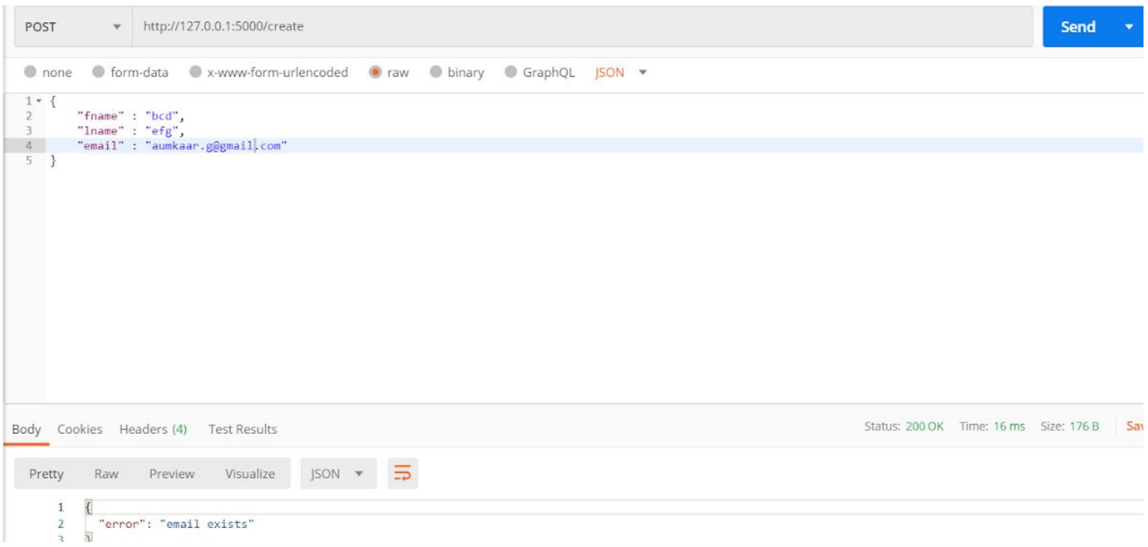
```
[
  [
    3,
    "bcd",
    "efg",
    "sample.com"
  ]
]
```

Updated database after adding :



```
sqlite> select * from user;
1|Aumkar|Gadekar|aumkaar.g@gmail.com
2|abc|skkks|xyz@gmail.com
sqlite> select * from user;
1|Aumkar|Gadekar|aumkaar.g@gmail.com
2|abc|skkks|xyz@gmail.com
3|bcd|efg|sample.com
sqlite> _
```

In case email already exists :



```
{
    "fname" : "bcd",
    "lname" : "efg",
    "email" : "aumkaar.g@gmail.com"
}
```

```
{
    "error": "email exists"
}
```

Add paths and run the app :

```
api.add_resource(byid, '/userbyid/<id>')
api.add_resource(all, '/all')

if __name__ == '__main__':
    app.run(debug=True)
```

Note : Flask, python and sqlite binaries will be required for the project to learn. APIs can be tested using Postman or with Curl from the command line, or through the browser window