

CRUD

1. DELETE

The screenshot displays a REST client interface with a dark theme. At the top, there's a navigation bar with 'Workspaces' and 'More' dropdowns, a search icon, a user icon, a settings gear, a bell, and an 'Upgrade' button. Below this, a tab bar shows three active requests: 'GET https://flaskweb-hft', 'POST http://localhost:5000', and 'DEL http://127.0.0.1:5000'. The selected request is a DELETE method to 'http://127.0.0.1:5000/users/1'. The interface includes a 'Send' button and a 'Share' button. Below the request bar, there are tabs for 'Params', 'Auth', 'Headers (9)', 'Body', 'Scripts', and 'Settings'. The 'Body' tab is active, showing a raw JSON request body:

```
1 {
2   "username": "balaji",
3   "email": "balaji@example.com",
4   "password": "balaji",
5   "name": "balaji"
6 }
```

 Below the request body, the response section shows a status of '200 OK' with a response time of '507 ms' and a size of '191 B'. The response body is also in JSON format:

```
1 {
2   "message": "User deleted successfully"
3 }
```

2. ADD USERS

The screenshot shows the Postman interface with a POST request to `http://127.0.0.1:5000/users`. The request body is a JSON object with the following fields:

```
{
  "username": "balaji",
  "email": "balaji@example.com",
  "password": "balaji",
  "name": "balaji"
}
```

The response is a 201 Created status, indicating successful registration. The response body is a JSON object:

```
{
  "message": "User registered successfully",
  "user": {
    "email": "balaji@example.com",
    "id": 2,
    "name": "balaji",
    "username": "balaji"
  }
}
```

The interface also shows a status bar at the bottom with the text "Almost done 🚀".

3. GET ALL USERS

The screenshot shows a REST client interface with a dark theme. At the top, there's a header bar with 'Workspaces' and 'More' dropdowns, a search icon, a user icon, a settings icon, a bell icon, and an 'Upgrade' button. Below this, a breadcrumb trail shows the request path: 'GET https://flaskweb-hfr' (disabled), 'POST http://localhost:5000' (disabled), and 'GET http://127.0.0.1:5000' (active). The main area is divided into two sections. The top section is for the request, showing the method 'GET' and the URL 'http://127.0.0.1:5000/users'. It includes tabs for 'Params', 'Auth', 'Headers (9)', 'Body', 'Scripts', and 'Settings'. The 'Body' tab is selected, showing a JSON payload in 'raw' mode:

```
{  "username": "ba",  "email": "bji2@example.com",  "password": "ba",  "name": "balaji"}
```

. The bottom section shows the response, with a status of '200 OK', a time of '252 ms', and a size of '280 B'. It also has tabs for 'JSON', 'Preview', and 'Visualize'. The 'JSON' tab is selected, showing the response body:

```
{  "users": [    {      "email": "bji2@example.com",      "id": 1,      "name": "balaji",      "username": "ba"    }  ]}
```

4. GET USERS BY ID

The screenshot shows the Postman interface for a REST client. The top bar displays the workspace name "Workspaces" and a search bar. The main area shows a GET request to the URL `http://127.0.0.1:5000/users/2`. The request is configured with Basic Auth, where the username is `balaji11` and the password is `balaji11`. The response status is `200 OK` with a response time of `229 ms` and a body size of `240 B`. The response body is displayed in JSON format, showing the user details for ID 2.

Menu GET https://flaskweb-hfr POST http://localhost:5000 GET http://127.0.0.1:5000 Upgrade No environment

HTTP http://127.0.0.1:5000/users/2 Save Share

GET http://127.0.0.1:5000/users/2 Send

Params Auth Headers (9) Body Scripts Settings Cookies

Auth Type

Basic Auth

Username balaji11

Password balaji11

The authorization header will be automatically generated when you send the request. Learn more about Basic Auth

Body 200 OK • 229 ms • 240 B

JSON Preview Visualize

```
1 {
2   "email": "balaji@example.com",
3   "id": 2,
4   "name": "balaji",
5   "username": "balaji"
6 }
```

5. UPDATE

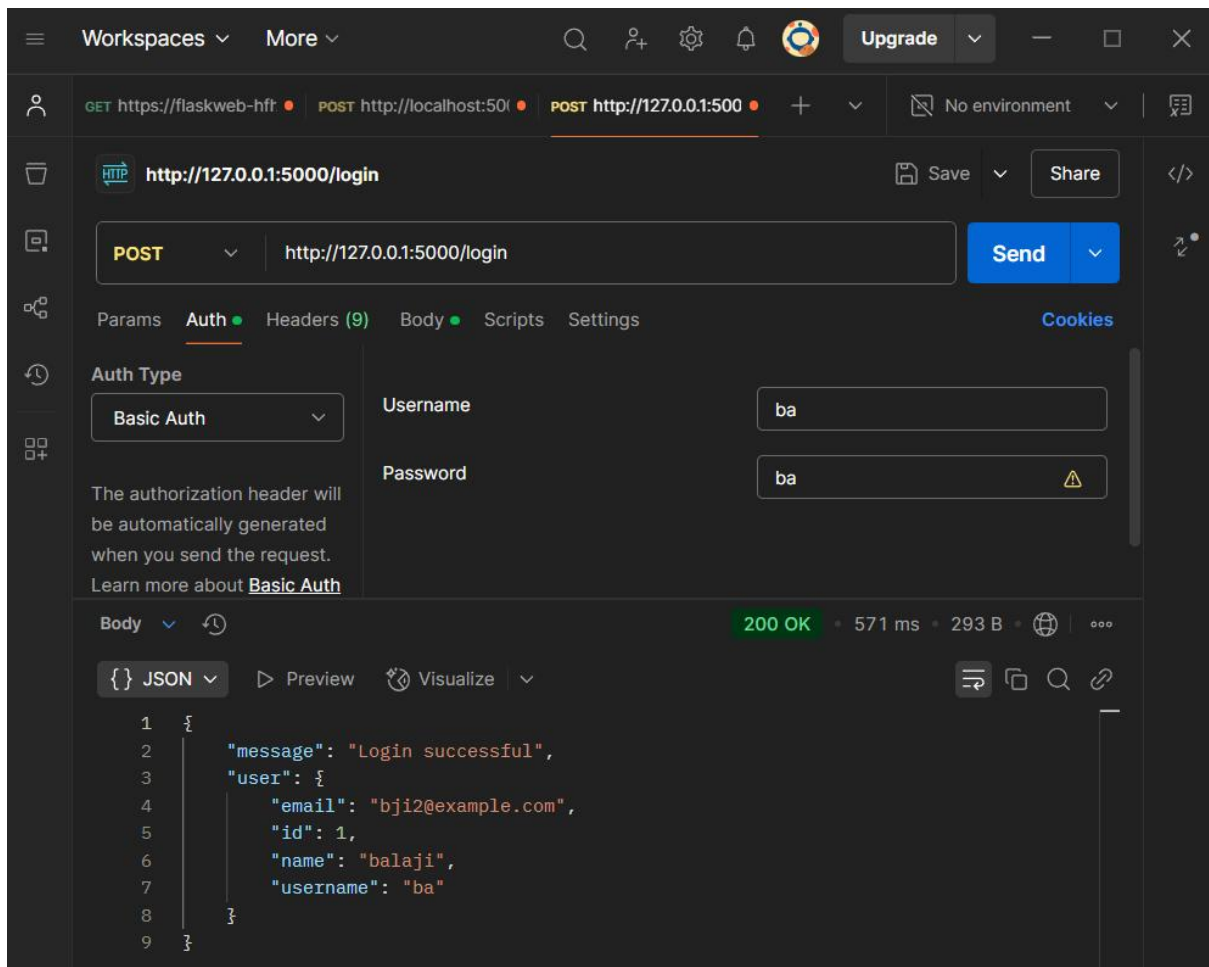
The screenshot displays the Postman interface for a PUT request. The URL bar shows `http://127.0.0.1:5000/users/3`. The request method is set to **PUT**. The request body is formatted as JSON and contains the following data:

```
1 {
2   "username": "balaji11",
3   "email": "balaji12345@example.com",
4   "password": "balaji11",
5   "name": "balaji"
6 }
```

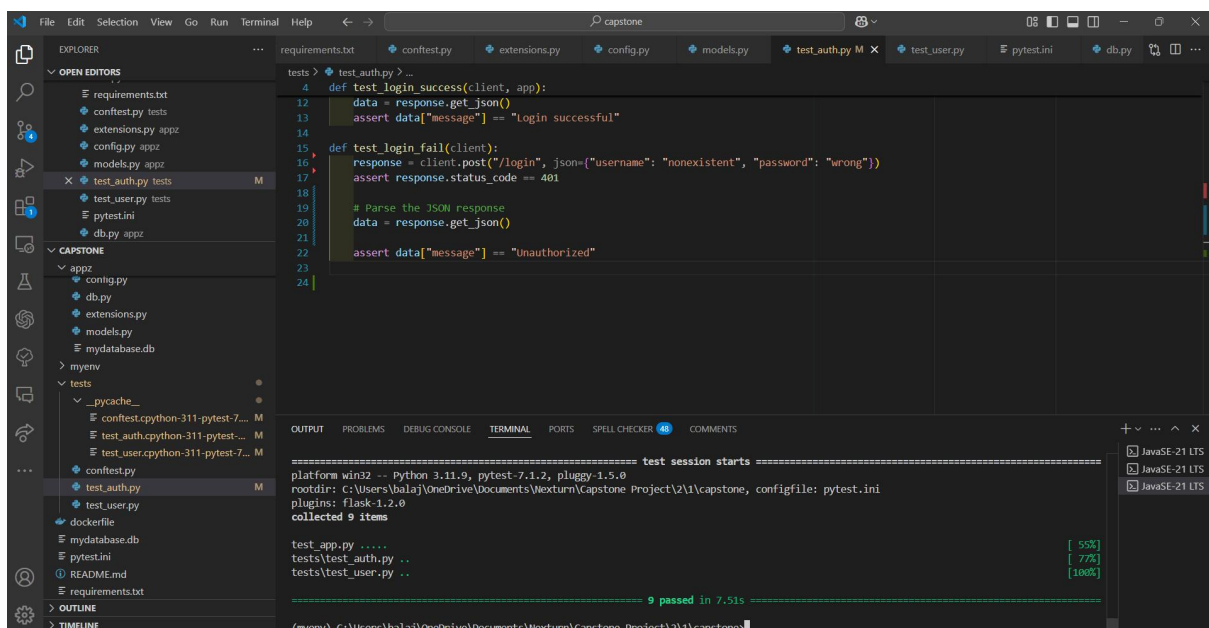
The response status is **200 OK**, with a response time of 989 ms and a body size of 315 B. The response body is also in JSON format:

```
1 {
2   "message": "User updated successfully",
3   "user": {
4     "email": "balaji12345@example.com",
5     "id": 3,
6     "name": "balaji",
7     "username": "balaji11"
8   }
9 }
```

AUTHENTICATE



TESTING



DOCKER

The screenshot shows the Docker Desktop interface. At the top, there's a search bar and a 'Ctrl+K' button. The main area displays the 'objective_cannon' container, which is running. The container's status is 'Exited (0) (43 minutes ago)'. The container's IP is 192.168.0.18, and it has a port 5000 open. The container's memory usage is 34.32% (1.341GiB / 3.906GiB) and its CPU usage is 2.97%.

The container's logs show the following output:

```
2025-02-20 22:42:47 * Running on all addresses.
2025-02-20 22:42:47 * Serving Flask app 'run.py' (lazy loading)
2025-02-20 22:42:47 * Environment: development
2025-02-20 22:42:47 * Debug mode: on
2025-02-20 22:42:47 WARNING: This is a development server. Do not use it in a production deployment.
2025-02-20 22:42:47 * Running on http://172.17.0.2:5000/ (Press CTRL+C to quit)
2025-02-20 22:42:47 * Restarting with stat
2025-02-20 22:42:48 * Debugger is active!
2025-02-20 22:42:48 * Debugger PIN: 519-880-376
```

Below the logs, there's a terminal window showing the command to run the container:

```
$ docker run -p 5000:5000 balajiyoganantham/flask-api-usermanagement
* Serving Flask app 'run.py' (lazy loading)
* Environment: development
* Debug mode: on
* Running on all addresses.
WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://172.17.0.2:5000/ (Press CTRL+C to quit)
* Restarting with stat
```