

## PyTest pass:

```
● (venv) PS C:\Users\antho\OneDrive\Desktop\Adv Web Dev Labs\devops_flask_activity> pytest -q
.....
5 passed in 1.34s
○ (venv) PS C:\Users\antho\OneDrive\Desktop\Adv Web Dev Labs\devops_flask_activity>
```

## Git Commits working:

```
(venv) PS C:\Users\antho\OneDrive\Desktop\Adv Web Dev Labs\devops_flask_activity> git commit -m "Lint issue fix"
[main bfd6530] Lint issue fix
 1 file changed, 4 insertions(+), 2 deletions(-)
(venv) PS C:\Users\antho\OneDrive\Desktop\Adv Web Dev Labs\devops_flask_activity> git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 617 bytes | 154.00 KiB/s, done.
Total 5 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/AnthonyCiceuOTU/devops_flask_activity.git
  35e62f6..bfd6530 main -> main
(venv) PS C:\Users\antho\OneDrive\Desktop\Adv Web Dev Labs\devops_flask_activity> pytest -q
```

## Actions on GitHub also pass:

The screenshot shows the GitHub Actions history for a repository. It displays 14 workflow runs. Three specific runs are highlighted under the 'Lint issue fix' category, each with a green checkmark icon. The first two runs are associated with 'DevOps Flask activity - CodeQL #4' and the third with 'DevOps Flask activity - Tests #4'. All three runs were pushed by 'AnthonyCiceuOTU' and were merged into the 'main' branch. The most recent run was performed 56 seconds ago at 2:42 PM. The other two runs were performed 16 seconds and 11 seconds ago respectively.

Event	Status	Branch	Actor	Time	...
DevOps Flask activity - CodeQL #4	Success	main	AnthonyCiceuOTU	Today at 2:42 PM	...
DevOps Flask activity - Tests #4	Success	main	AnthonyCiceuOTU	Today at 2:42 PM	...
DevOps Flask activity - Lint #4	Success	main	AnthonyCiceuOTU	Today at 2:42 PM	...

This is optional but I decided to test in POSTMAN too to make sure everything was working fine:

GET:

The screenshot shows the Postman interface with a dark theme. At the top, the URL `http://127.0.0.1:5000/hello` is entered in the address bar. Below it, a **GET** method is selected. The response status is **200 OK** with a duration of **38 ms** and a size of **198 B**. The response body is JSON, containing the message `"Hello, World!"`.

## POST:

The screenshot shows the Postman interface with a dark theme. A new request is being prepared to `http://127.0.0.1:5000/echo` using the **POST** method. The body is set to **raw** JSON, containing the payload `{"msg": "ping", "user": "Anthony"}`. The response status is **201 CREATED** with a duration of **56 ms** and a size of **211 B**.

## PUT:

The screenshot shows the Postman application interface. At the top, the URL `http://127.0.0.1:5000/items/1` is entered in the address bar. Below it, the method is set to `PUT` and the target URL is again `http://127.0.0.1:5000/items/1`. The `Body` tab is selected, showing the raw JSON payload:

```
1 {  
2   "name": "example item",  
3   "value": 42  
4 }
```

The `Send` button is highlighted in blue. The response section shows a `200 OK` status with `10 ms`, `240 B`, and a globe icon indicating international reach. The response body is displayed as:

```
{ } JSON ▾ > Preview Visualize ▾
```

```
1 {  
2   "data": {  
3     "name": "example item",  
4     "value": 42  
5   },  
6   "id": 1  
7 }
```

The bottom of the screen shows the Windows taskbar with various pinned icons and system status indicators like battery level, signal strength, and the date/time `9:29 AM 2025-11-17`.

## DELETE:

The screenshot shows the Postman application interface. At the top, the URL `http://127.0.0.1:5000/items/1` is entered into the address bar. Below the address bar, the method is set to `DELETE`. To the right of the address bar are buttons for `Save`, `Share`, and a copy icon. A large blue `Send` button is located on the right side of the header.

The main interface has tabs for `Params`, `Authorization`, `Headers (7)`, `Body` (which is currently selected), `Scripts`, and `Settings`. Below these tabs, there are several input options: `none`, `form-data`, `x-www-form-urlencoded`, `raw` (which is selected), `binary`, `GraphQL`, and `JSON`. There is also a `Beautify` button.

In the body section, there is a code editor containing the following JSON:

```
1  Ctrl+Alt+P to Ask AI
2
3
4
5
6
7
```

Below the code editor, the status bar shows `200 OK`, `8 ms`, `243 B`, and a globe icon. The bottom navigation bar includes links for `Postbot`, `Runner`, `Start Proxy`, `Cookies`, `Vault`, `Trash`, and a help icon.

Trying to **DELETE** again but with no item there:

The screenshot shows the Postman application interface. At the top, the URL `http://127.0.0.1:5000/items/1` is entered into the address bar. Below it, the method is set to **DELETE**. The **Body** tab is selected, showing the raw JSON response: 

```
1  {
2   |   "error": "Item not found"
3 }
```

 The status bar at the bottom indicates a **404 NOT FOUND** response with **16 ms** duration and **204 B** size.