

CLOUD COMPUTING

LAB 2 - MONOLITHIC ARCHITECTURE

NAME: SOWMYA HARDAGERI

SRN: PES2UG23CS590

SECTION: I

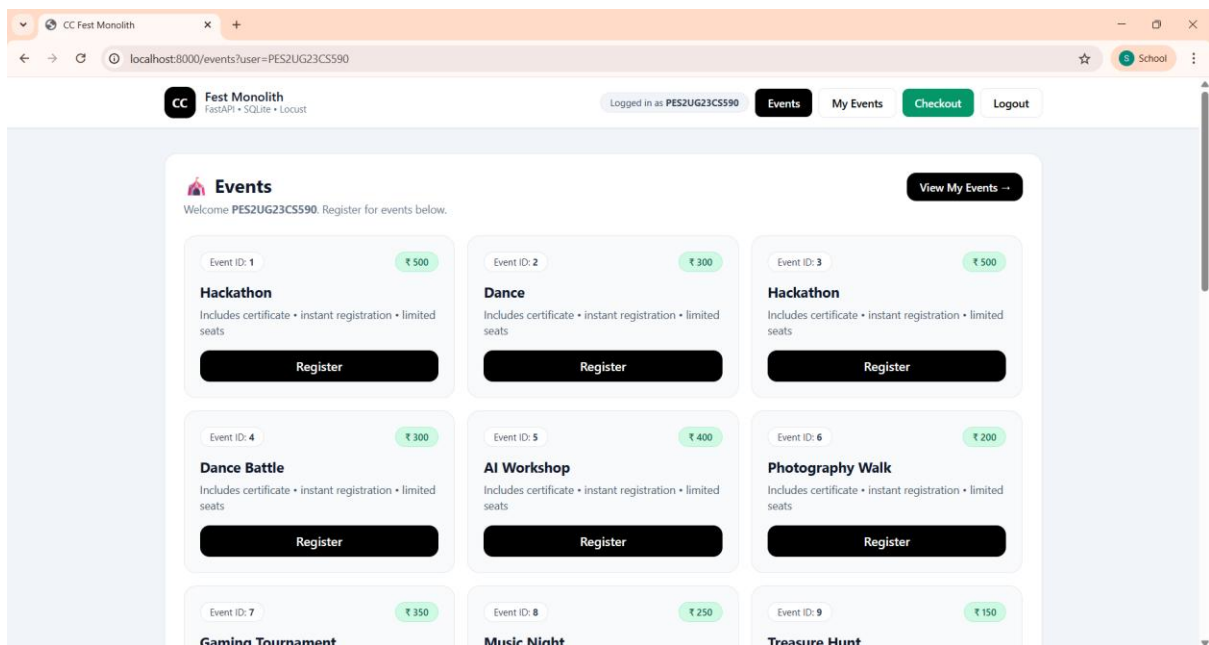
CAMPUS: ELECTRONIC CITY CAMPUS

GITHUB LINK: https://github.com/SowmyaHpes2ug23cs590/CC_lab2

Screenshots:

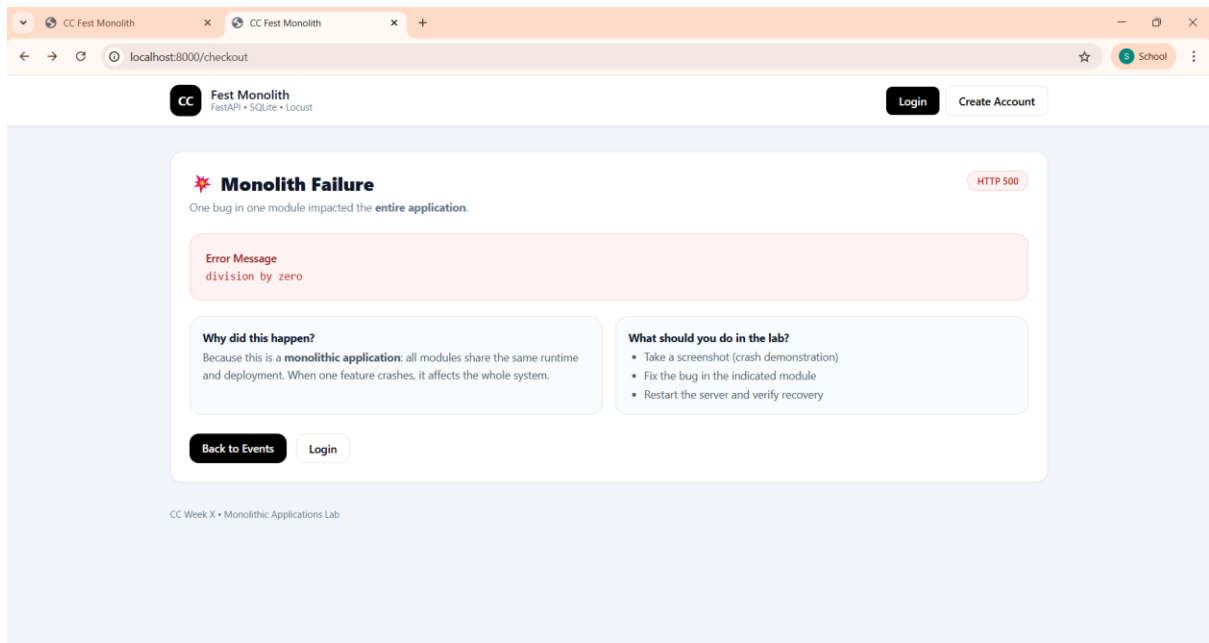
PART 2: Use the Application

Account Created Successfully



PART 3: Observe Monolithic Failure (Crash)

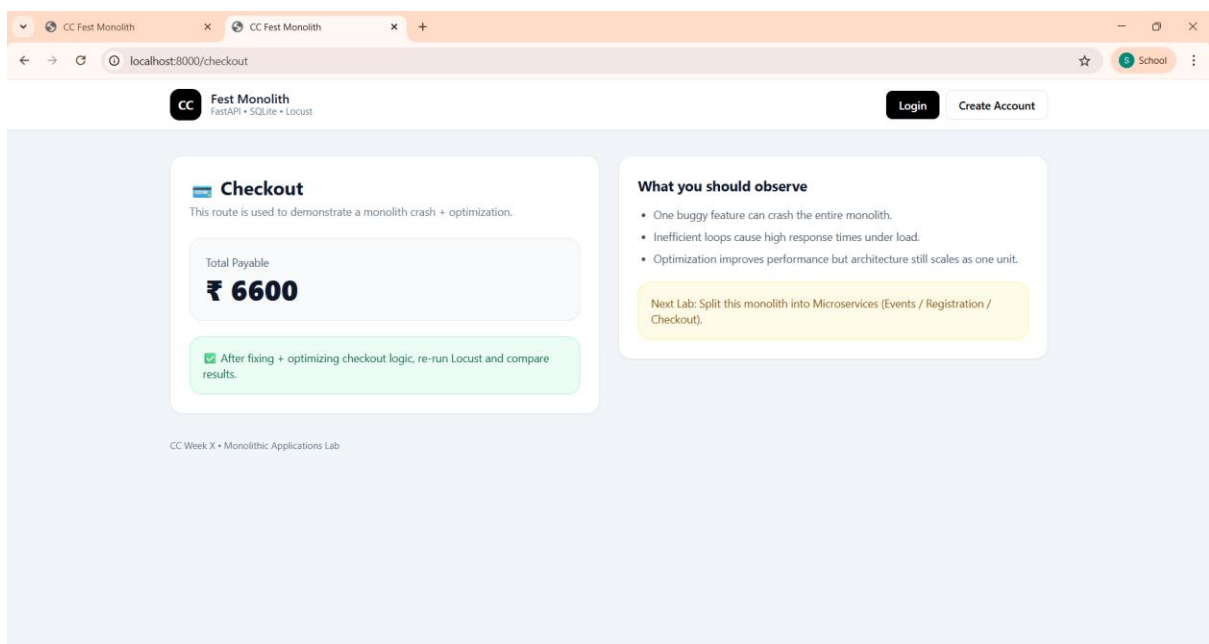
Monolithic Failed Successfully

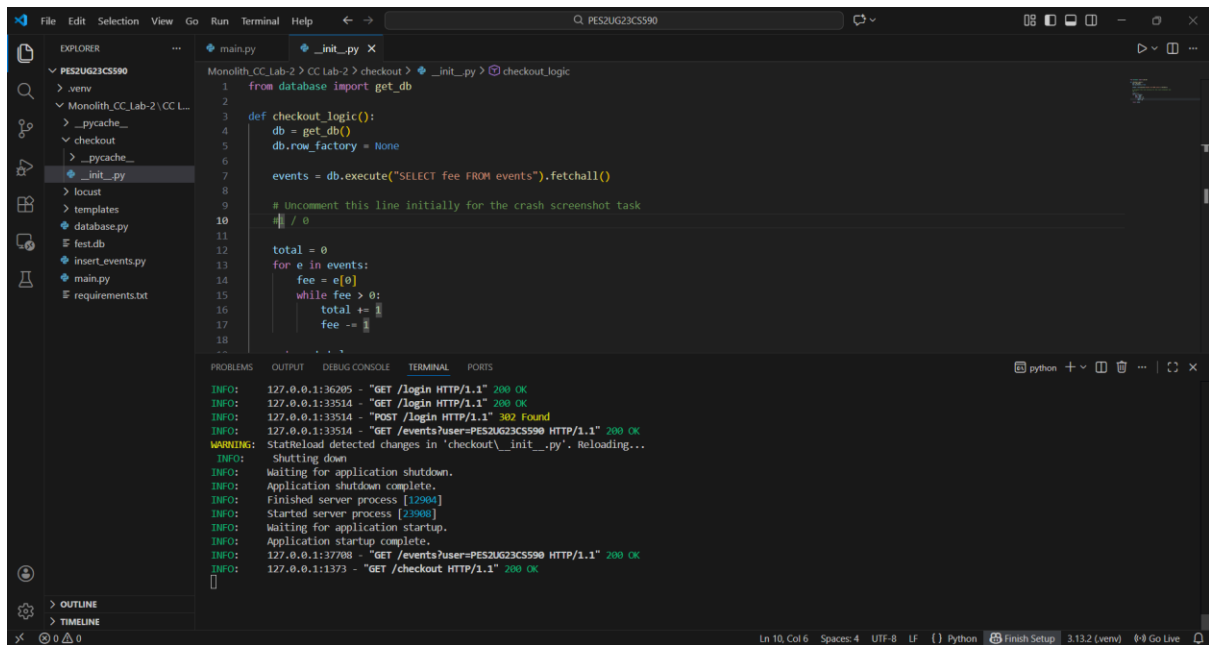


```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
INFO: 127.0.0.1:53746 - "POST /register HTTP/1.1" 302 Found
INFO: 127.0.0.1:53746 - "GET /login HTTP/1.1" 200 OK
INFO: 127.0.0.1:61606 - "POST /login HTTP/1.1" 302 Found
INFO: 127.0.0.1:61606 - "GET /events?user=PES2UG23CS590 HTTP/1.1" 200 OK
INFO: 127.0.0.1:1669 - "GET /events?user=PES2UG23CS590 HTTP/1.1" 200 OK
INFO: 127.0.0.1:32662 - "GET /checkout HTTP/1.1" 200 OK
WARNING: StatReload detected changes in 'checkout\_init_.py'. Reloading...
INFO: Shutting down
INFO: Waiting for application shutdown.
INFO: Application shutdown complete.
INFO: Finished server process [1954]
INFO: Started server process [12904]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: 127.0.0.1:53157 - "GET /checkout HTTP/1.1" 500 Internal Server Error
ERROR: Exception in ASGI application
Traceback (most recent call last):
  File "E:\Somya\PEP\SEPESTER o\CLOUD COMPUTING\LAB\PEP2UG23CS590\.venv\Lib\site-packages\uvicorn\protocols\http\h11_impl.py", line 410, in run_asgi
    Ln 10, Col 6 Spaces: 4 UTF-8 LF Python Finish Setup 3.13.2 (venv) 0-0 Go Live
```

PART 4: Fix the Bug

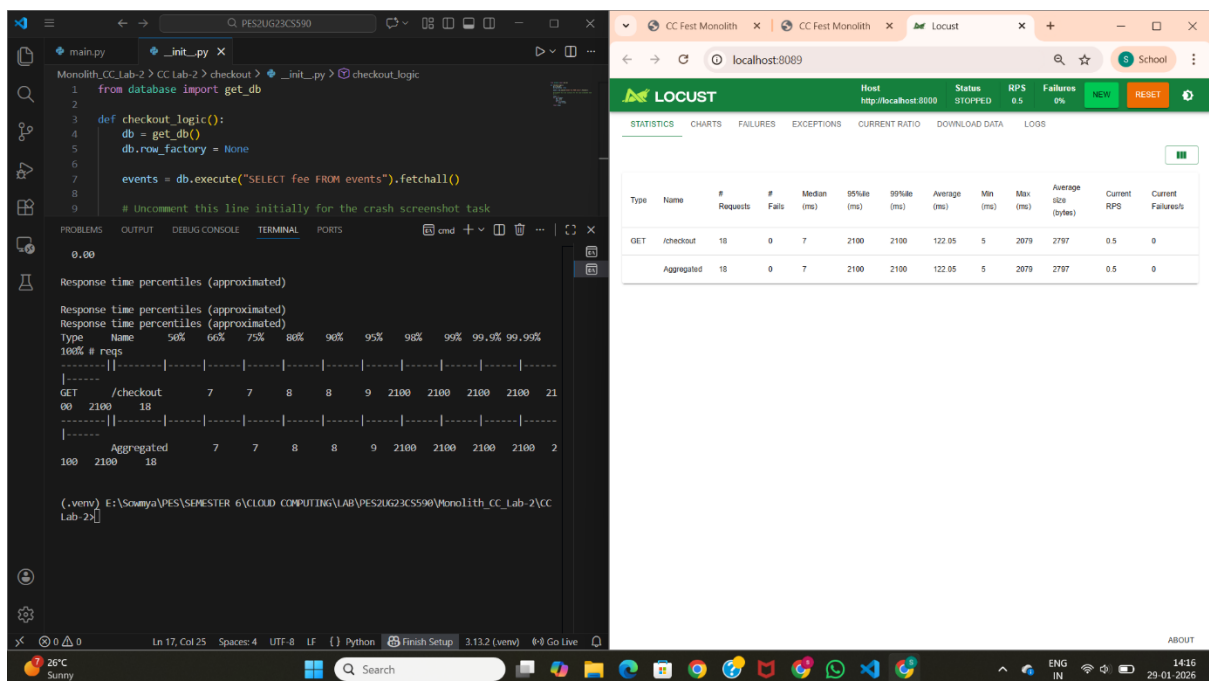
Bug Fixed





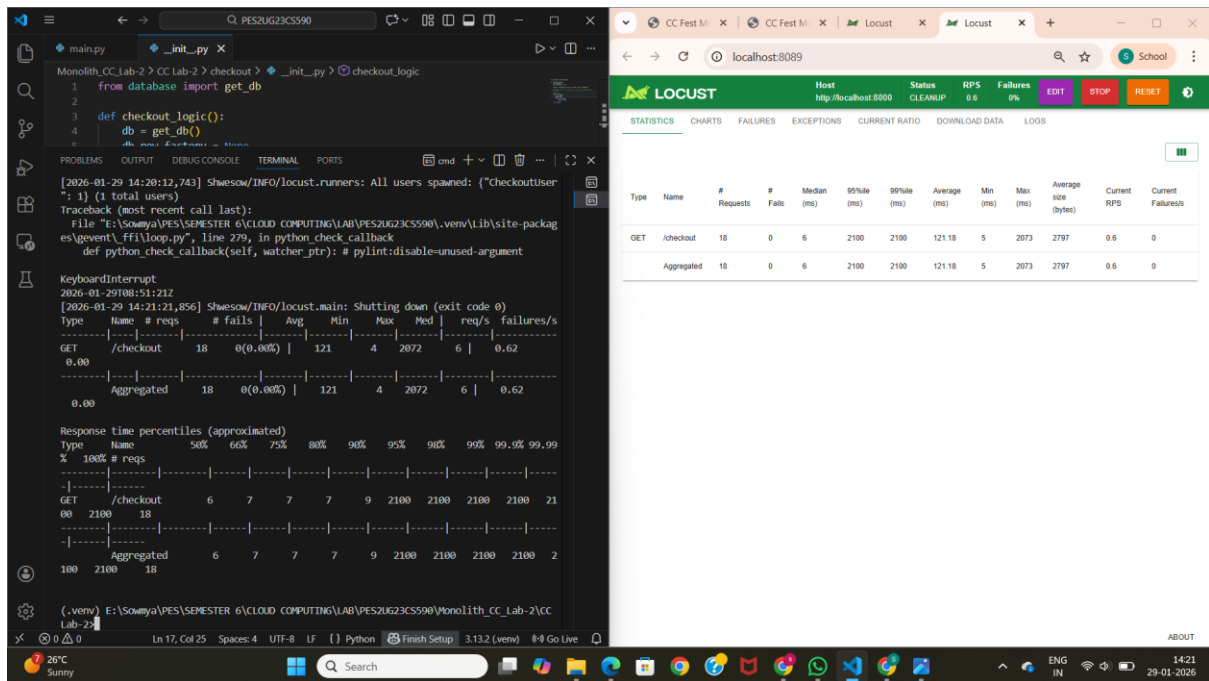
PART 5: Load Testing using Locust

Terminal and Locust before Optimization



PART 6: Optimize the Checkout Route

Terminal and Locust after Optimization

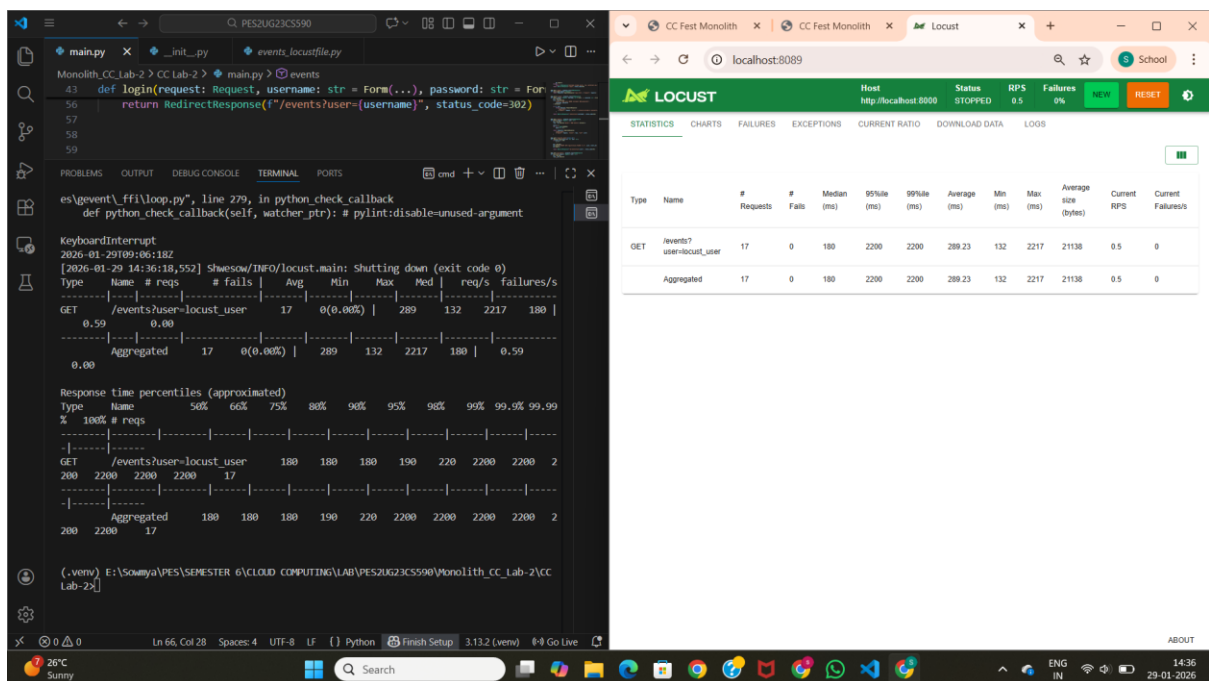


Observation: Average response time before optimization was 122.05, after optimizing the code, the time reduced to 121.18 whereas RPS stayed almost the same.

PART 7: Optimise events and my_events(DIY)

Route 1: /events

Before Optimization:



After Optimization:

The screenshot shows the VS Code editor with the file `events_locustfile.py` open. The code defines a Locust user class `events` that sends a `GET` request to `/events?user=locust_user`. The terminal displays the output of the Locust load test, showing a significant improvement in performance compared to the 'Before Optimization' state.

Locust Statistics (After Optimization):

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/events?user=locust_user	19	0	6	2100	2100	114.42	3	2064	2138	0.6	0
Aggregated		19	0	6	2100	2100	114.42	3	2064	2138	0.6	0

Route 2: /my-events

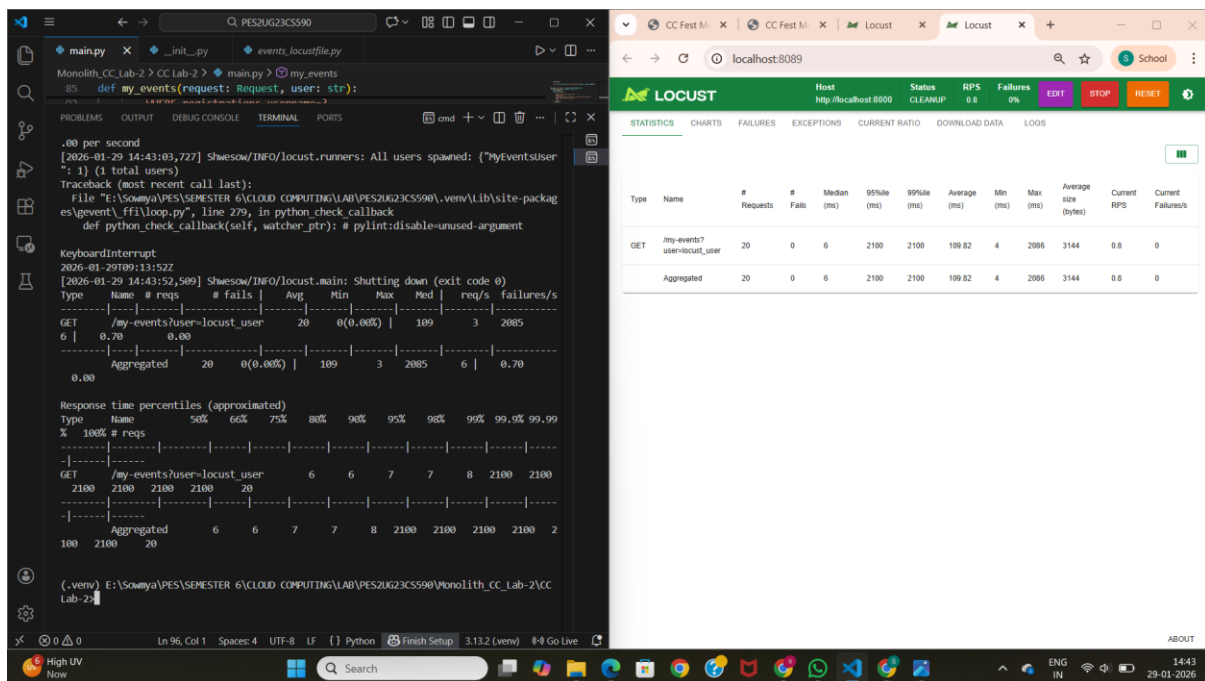
Before Optimization:

The screenshot shows the VS Code editor with the file `events_locustfile.py` open. The code defines a Locust user class `events` that sends a `GET` request to `/my-events?user=locust_user`. The terminal displays the output of the Locust load test, showing a high number of requests and a significant number of failures, indicating poor performance before optimization.

Locust Statistics (Before Optimization):

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/my-events?user=locust_user	18	0	69	2300	2300	190.26	57	2258	3144	0.7	0
Aggregated		18	0	69	2300	2300	190.26	57	2258	3144	0.7	0

After Optimization:



Route 1: /events

Performance Bottleneck:

The endpoint contained an artificial CPU-heavy loop (for i in range(3000000)) that performed no meaningful work, consuming processing time and delaying request handling.

Optimization Implemented:

The unnecessary computation loop was completely removed from the request flow.

Reason for Performance Improvement:

Removing the CPU-bound operation significantly reduced execution time and allowed the server to process concurrent requests more efficiently.

Route 2: /my-events

Performance Bottleneck:

After fetching data from the database, the route executed a redundant loop (for _ in range(1500000)), introducing avoidable CPU overhead.

Optimization Implemented:

The non-functional loop was eliminated to streamline the request execution path.

Reason for Performance Improvement:

With the extra computation removed, the endpoint delivered faster responses and improved overall system throughput.