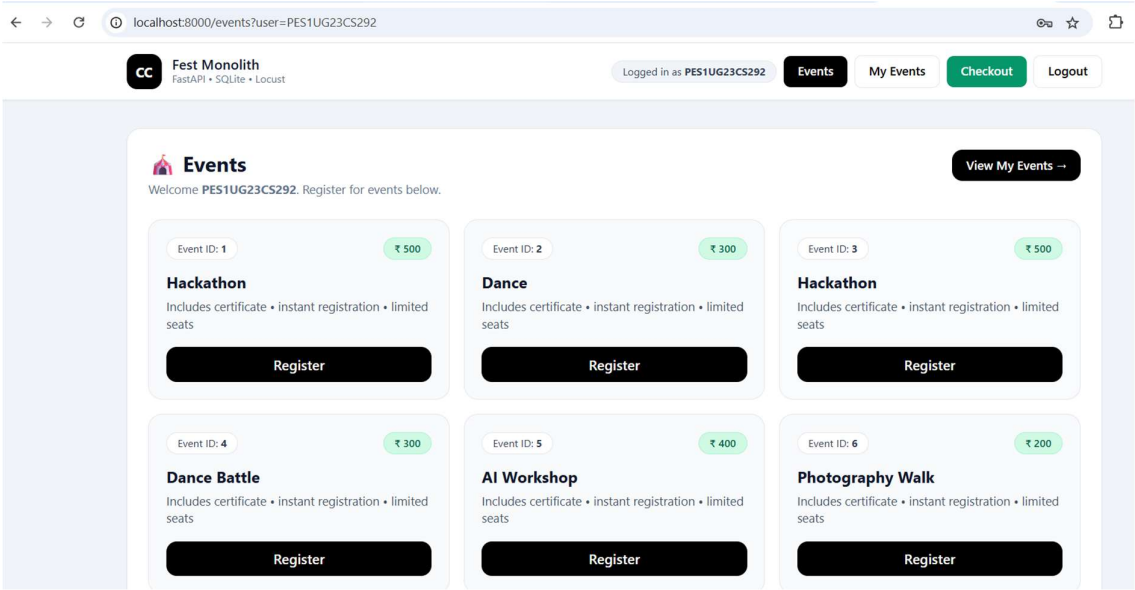


NAME:Kavana H

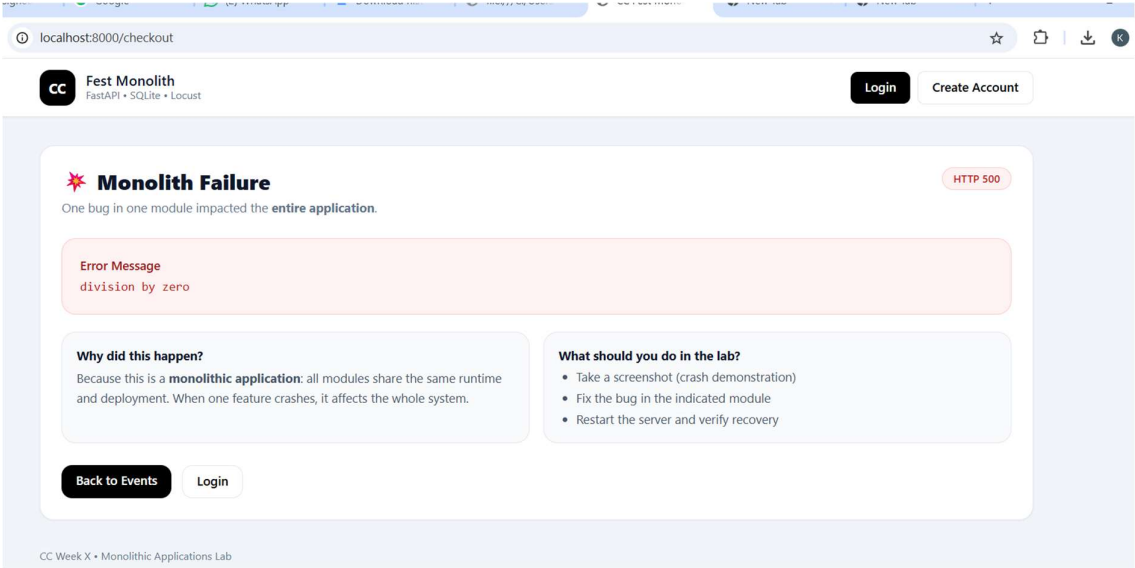
SRN:PES1UG23CS292

CCLAB2

SS1(EVENT PAGE LOADING):



SS2(CHECKOUT PAGE):



```
File "D:\SEM6\CCLAB\CCLAB2\PES1UG23CS292\CC Lab-2\main.py", line 113, in checkout
    total = checkout_logic()
    ~~~~~
File "D:\SEM6\CCLAB\CCLAB2\PES1UG23CS292\CC Lab-2\checkout\_init_.py", line 10, in checkout_logic
    1 / 0
    ~~~~
ZeroDivisionError: division by zero
INFO: 127.0.0.1:54418 - "GET /login HTTP/1.1" 200 OK
INFO: 127.0.0.1:54418 - "POST /login HTTP/1.1" 302 Found
INFO: 127.0.0.1:54418 - "GET /events?user=PES1UG23CS292 HTTP/1.1" 200 OK
INFO: 127.0.0.1:57598 - "GET /checkout HTTP/1.1" 500 Internal Server Error
ERROR: Exception in ASGI application
Traceback (most recent call last):
```

SS3(AFTER FIXING BUG):

localhost:8000/checkout

cc

Fest Monolith
FastAPI • SQLite • Locust

LoginCreate Account

Checkout

This route is used to demonstrate a monolith crash + optimization.

Total Payable

₹ 6600

After fixing + optimizing checkout logic, re-run Locust and compare results.

What you should observe

- One buggy feature can crash the entire monolith.
- Inefficient loops cause high response times under load.
- Optimization improves performance but architecture still scales as one unit.

Next Lab: Split this monolith into Microservices (Events / Registration / Checkout).

CC Week X • Monolithic Applications Lab

```
PS D:\SEM6\CCLAB\CCBLAB2\PES1UG23CS292\CC Lab-2> & D:/SEM6/CCLAB/CCBLAB2/PES1UG23CS292/.venv/Scripts/Activate.ps1
(.venv) PS D:\SEM6\CCLAB\CCBLAB2\PES1UG23CS292\CC Lab-2> uvicorn main:app --reload
INFO: Will watch for changes in these directories: ['D:\SEM6\CCLAB\CCBLAB2\PES1UG23CS292\CC Lab-2']
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO: Started reloader process [12232] using StatReload
INFO: Started server process [2892]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: 127.0.0.1:51959 - "GET /checkout HTTP/1.1" 200 OK
```

SS4(LOADING USING LOCUST):

Locust

Monolithic Applications_stud

localhost:8089

LOCUST

STATISTICSCHARTSFAILURES EXCEPTIONSCURRENT RATIO

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	M (r
GET	/checkout	18	0	9	2100	2100	122.78	4
Aggregated		18	0	9	2100	2100	122.78	4

Locust

Monolithic Applications_stud

localhost:8089

LOCUST

STATISTICSCHARTSFAILURES EXCEPTIONSCURRENT RATIO

Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
2100	2100	2100	122.78	4	2064	2797	0.7	0
2100	2100	2100	122.78	4	2064	2797	0.7	0

main.py

__init__.py

CC Lab-2 > checkout > __init__.py > checkout_logic

1 from database import get_db

PROBLEMS

OUTPUT

TERMINAL

PORTS

pwhsh - CC Lab-2

+

+

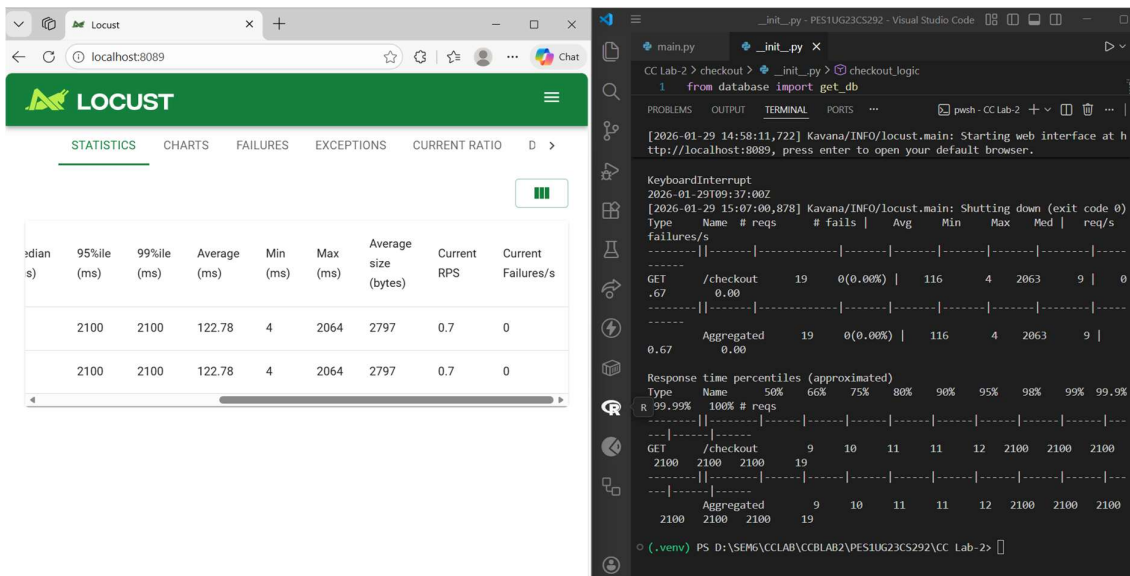
+

+

[2026-01-29 14:58:11,722] Kavana/INFO/locust.main: Starting web interface at h
http://localhost:8089, press enter to open your default browser.

Type	Name	50%	66%	75%	80%	90%	95%	98%	99%	99.9%
GET	/checkout	9	10	11	11	12	2100	2100	2100	2100
Aggregated		9	10	11	11	12	2100	2100	2100	2100

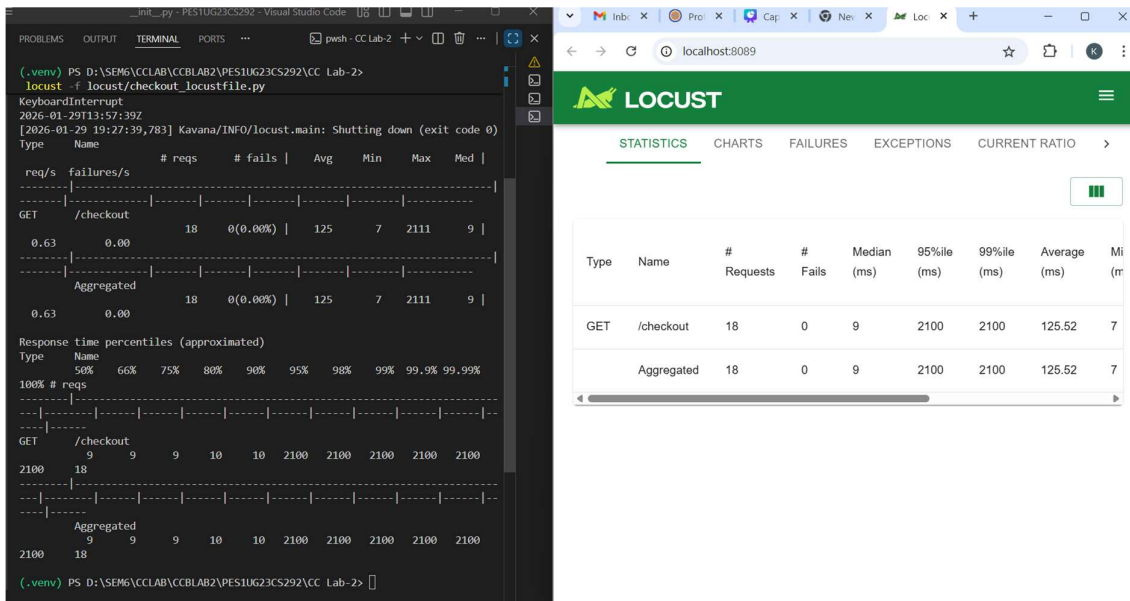
(.venv) PS D:\SEM6\CCLAB\CCBLAB2\PES1UG23CS292\CC Lab-2>

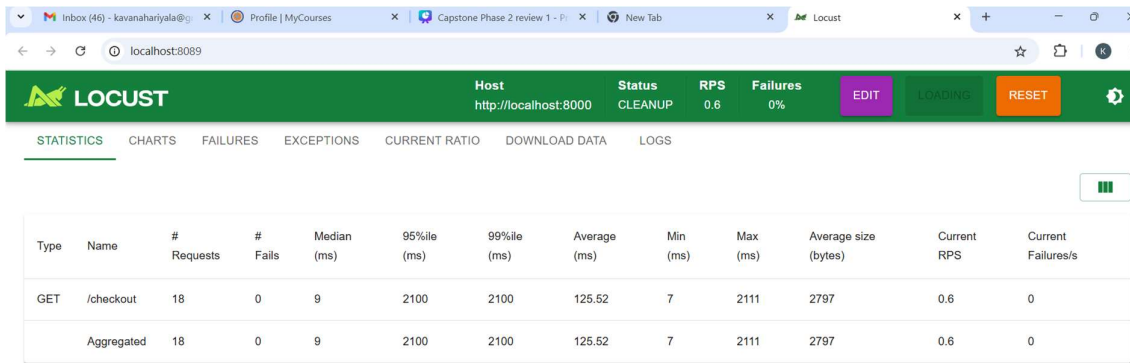


SS5(after optimization):

OPTIMIZATION DONE IS REMOVED THE WHILE LOOP

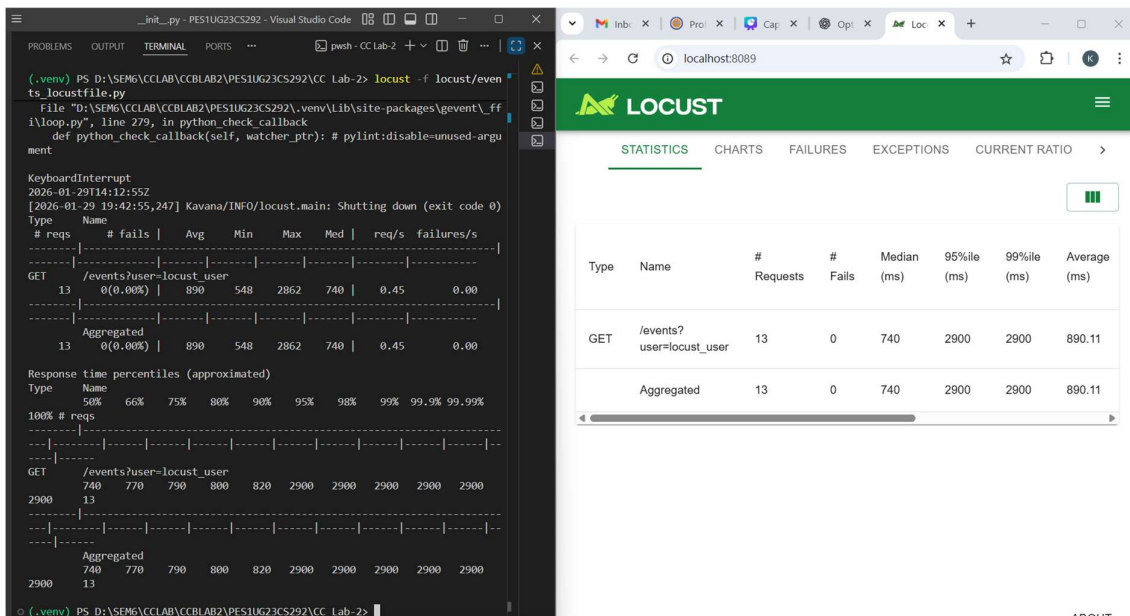
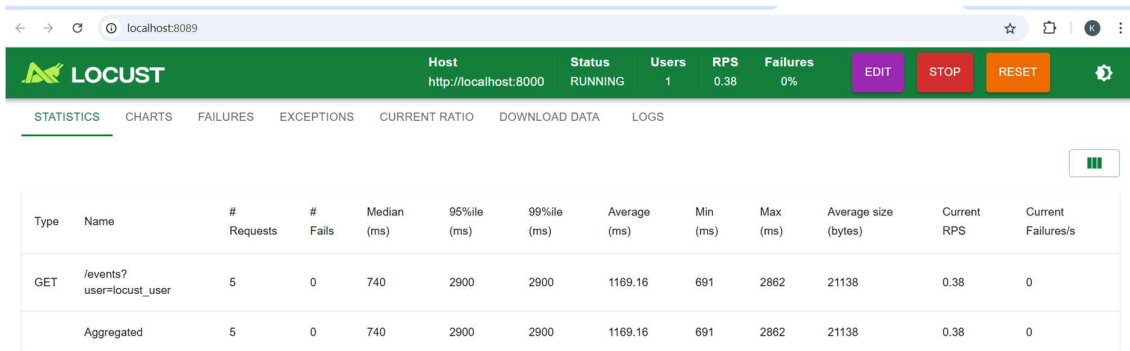
The original checkout logic had an $O(n \times \text{fee})$ time complexity due to a nested loop that incremented per unit fee. This was optimized to $O(n)$ by summing fees directly, and further improved to $O(1)$ at the application level by using SQL aggregation (SUM), significantly improving performance under concurrent load.





SS6(unoptimized events):

locust:



localhost:8089

<

SS7(events optimized):

Bottleneck:

A loop running 3,000,000 iterations was wasting CPU time on every request.

Change made:

Removed the unnecessary loop since it did not affect application logic.

Why performance improved:

The server no longer spends time on useless computation, reducing response time and allowing it to handle more requests efficiently.

main.py - PES1UG23CS292 - Visual Studio Code 08

PROBLEMS OUTPUT TERMINAL PORTS

push - CC Lab-2

Type Name # reqs # fails | Avg Min Max Med | req/s failures/s

[2026-01-29 19:50:18,853] Kavana/INFO/locust.main: Shutting down (exit code 0)

Type Name # reqs # fails | Avg Min Max Med | req/s failures/s

GET /events?user=locust_user 20 0(0.00%) | 114 8 210

5 9 | 0.67 0.00

Aggregated 20 0(0.00%) | 114 8 210 9 |

0.67 0.00

Response time percentiles (approximated)

Type Name 50% 66% 75% 80% 90% 95% 98% 99% 99.9%

99.99% # reqs

GET /events?user=locust_user 10 11 11 11 13 2100

2100 2100 2100 2100 20

Aggregated 10 11 11 11 13 2100 2100 2100

2100 2100 20

(.venv) PS D:\SEM6\CCLAB2\PES1UG23CS292\CC Lab-2>

localhost8089

LOCUST

STATISTICS CHARTS FAILURES EXCEPTIONS CURRENT RATIO

Type Name # Requests # Fails Median (ms) 95%ile (ms) 99%ile (ms) Average (ms)

GET /events?user=locust_user 20 0 9 2100 2100 114.84

Aggregated 20 0 9 2100 2100 114.84

localhost:8089

LOCUST

Host

http://localhost:8000

Status

STOPPED

RPS

0.7

Failures

0%

NEW

RESET

STATISTICS

CHARTS

FAILURES

EXCEPTIONS

CURRENT RATIO

DOWNLOAD DATA

LOGS

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	20	0	9	2100	2100	114.84	9	2106	21138	0.7	0
	Aggregated	20	0	9	2100	2100	114.84	9	2106	21138	0.7	0

SS8(my_events initial file):

LOCUST

Host
http://localhost:8000

Status
CLEANUP

RPS
0.6

Failures
0%

EDIT

STOP

RESET

STATISTICS

CHARTS

FAILURES

EXCEPTIONS

CURRENT RATIO

DOWNLOAD DATA

LOGS

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	16	0	240	2300	2300	365.69	194	2305	3144	0.6	0
	Aggregated	16	0	240	2300	2300	365.69	194	2305	3144	0.6	0

main.py - PESTUG23CS292 - Visual Studio Code

main.py

database.py

init.py

CC Lab-2 > main.py > register_event

43 f login(request: Request, username: str = Form(...), password: str =

PROBLEMS OUTPUT TERMINAL PORTS

pwsh - CC Lab-2

[2026-01-29 19:50:18,853] Kavana/INFO/locust.main: Shutting down (exit code 0)

Type # reqs # fails | Avg Min Max Med | req/s failures/s

GET /my-events?user=locust_user 16 0(0.00%) | 365 194 2305 240 | 0.57

Aggregated 16 0(0.00%) | 365 194 2305 240 | 0.57

Response time percentiles (approximated)

Type Name 50% 66% 75% 80% 90% 95% 98% 99% 99.9% 99.99%

GET /my-events?user=locust_user 250 260 260 260 280 2300 2300 2300 2300

Aggregated 250 260 260 260 280 2300 2300 2300 2300

PS D:\SEM6\CCLAB\CCBLAB2\PESTUG23CS292\CC Lab-2>

localhost:8089

LOCUST

STATISTICS CHARTS FAILURES EXCEPTIONS CURRENT RATIO

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)
GET	/my-events?user=locust_user	16	0	240	2300	2300	365.69
	Aggregated	16	0	240	2300	2300	365.69

localhost:8089

LOCUST

Host
http://localhost:8000

Status
CLEANUP

RPS
0.6

Failures
0%

EDIT

LOADING

RESET

STATISTICS

CHARTS

FAILURES

EXCEPTIONS

CURRENT RATIO

DOWNLOAD DATA

LOGS

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	16	0	240	2300	2300	365.69	194	2305	3144	0.6	0
	Aggregated	16	0	240	2300	2300	365.69	194	2305	3144	0.6	0

SS9(MY EVENTS OPTIMIZED):

Bottleneck:

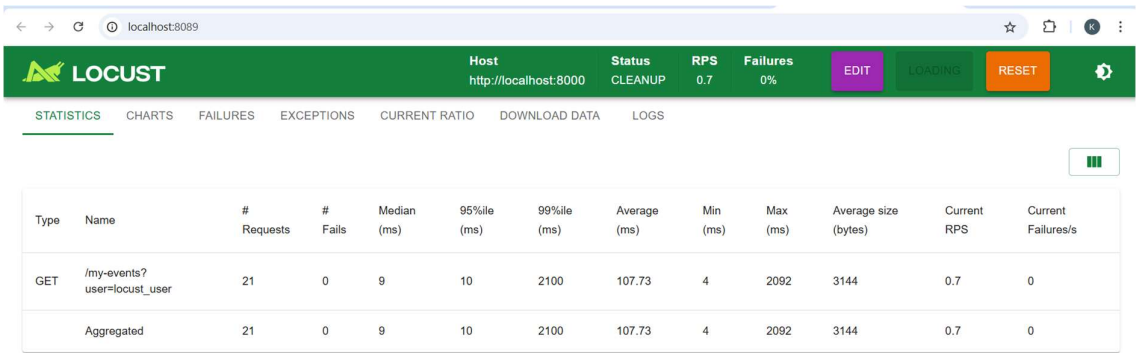
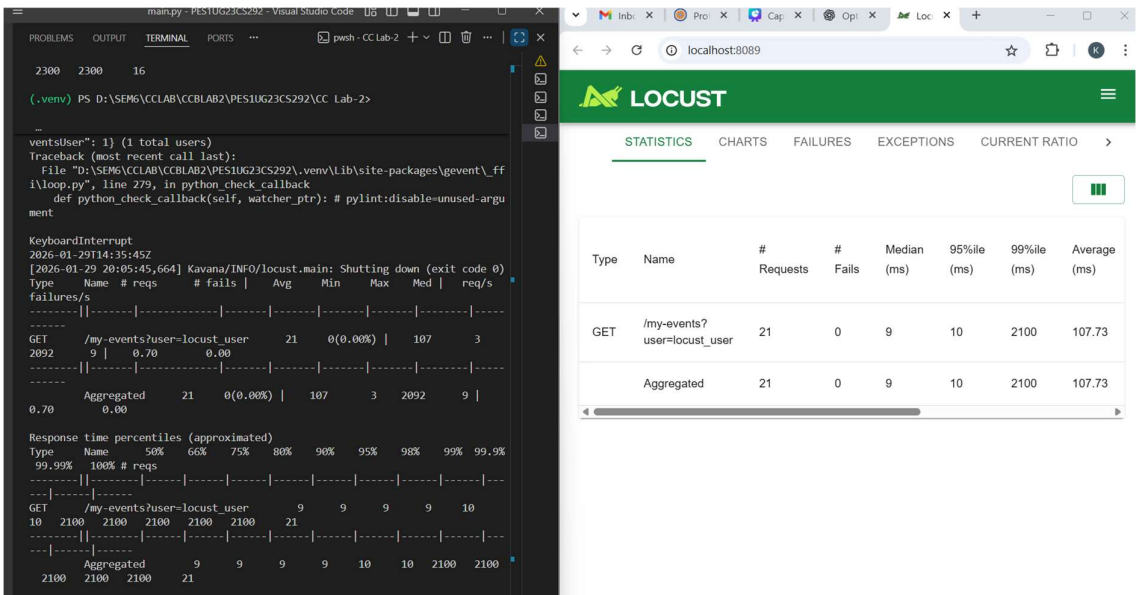
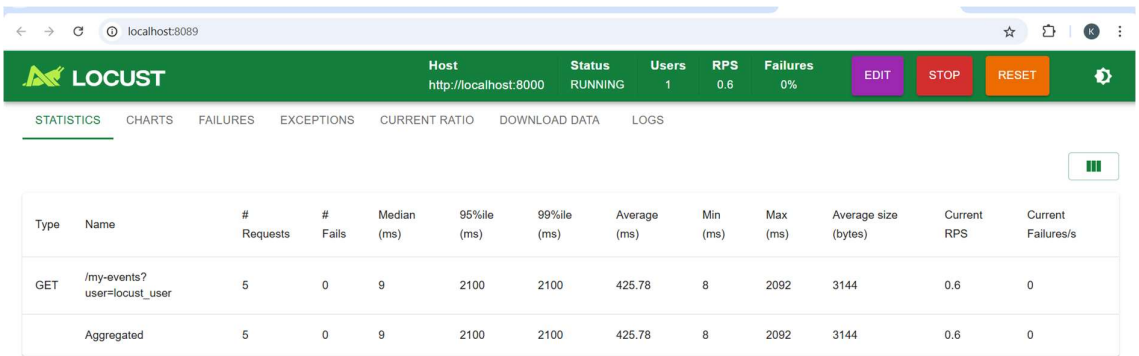
A loop with 1,500,000 iterations was unnecessarily consuming CPU resources for every request.

Change made:

Removed the loop since it did not contribute to the result of the route.

Why performance improved:

By eliminating the artificial computation, the route now responds faster and can handle more concurrent requests.



1. Checkout Endpoint (/checkout) – SS5

Performance Issue: The total fee was being calculated using an unnecessarily slow iterative approach.

Modification Done: Replaced the manual accumulation logic with a more efficient built-in summation method.

Result: The endpoint now responds quicker while maintaining the same ability to handle incoming requests.

2. Events Endpoint (/events) – SS7

Performance Issue: A large loop performing redundant calculations was increasing the response time.

Modification Done: Eliminated the loop since it had no impact on the actual output.

Result: The page loads significantly faster and CPU usage during requests is reduced.

3. My Events Endpoint (/my-events) – SS9

Performance Issue: An artificial loop was present that consumed processing time without contributing to functionality.

Modification Done: Removed this unnecessary computation from the route.

Result: The route now executes more efficiently and returns results with improved speed.