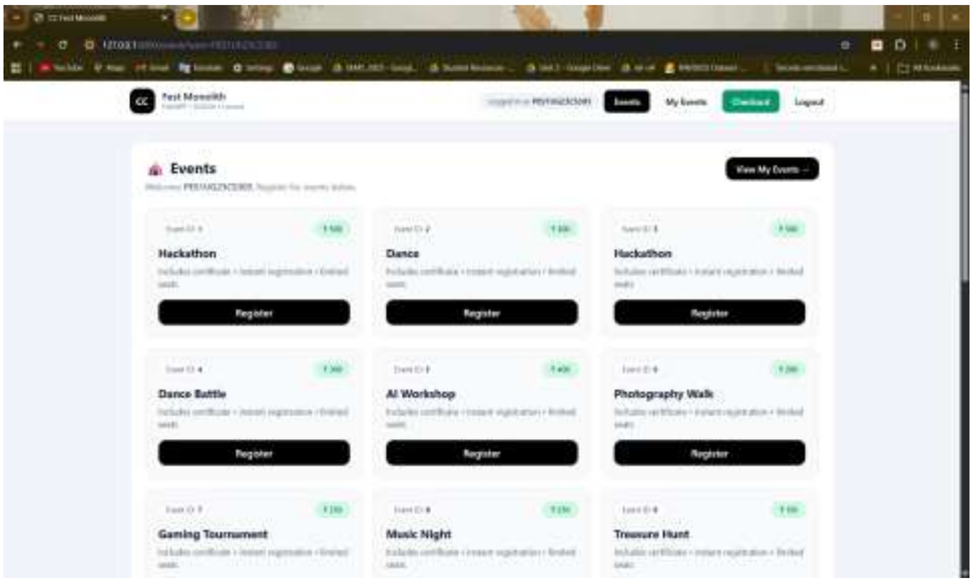
CC LAB 2

Name: Khushee P Kiran

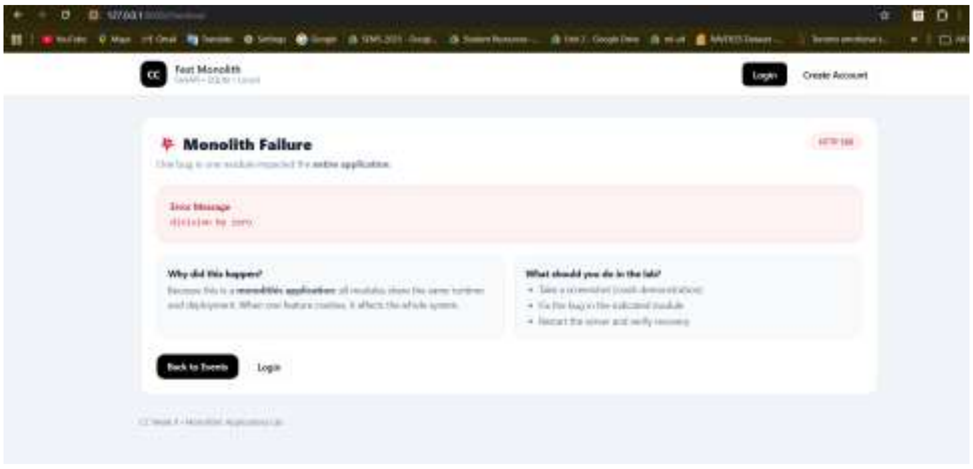
SEC: E

SRN: PES1UG23CS303

PART 2: Use the Application



PART 3: Observe Monolithic Failure (Crash)



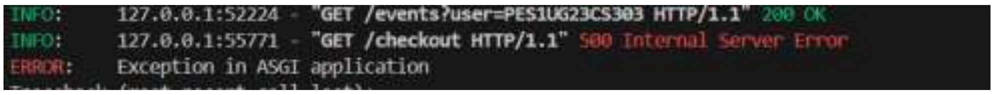
Monolith Failure

Error Message

division by zero

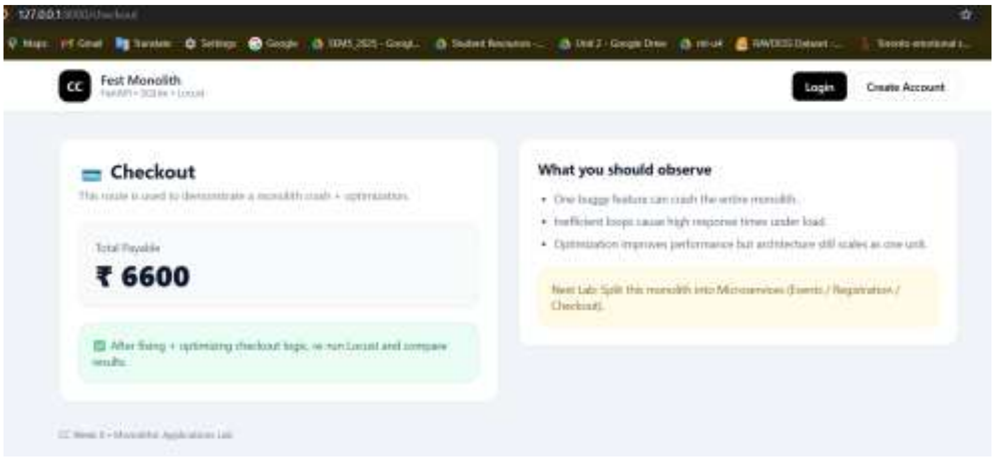
Why did this happen?

What should you do in the lab?



```
INFO:     127.0.0.1:52224 - "GET /events?user=PES1UG23CS303 HTTP/1.1" 200 OK
INFO:     127.0.0.1:55771 - "GET /checkout HTTP/1.1" 500 Internal Server Error
ERROR:    Exception in ASGI application
```

PART 4: Fix the Bug



Fest Monolith
FastAPI • SQLite • Locust

Login Create 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.

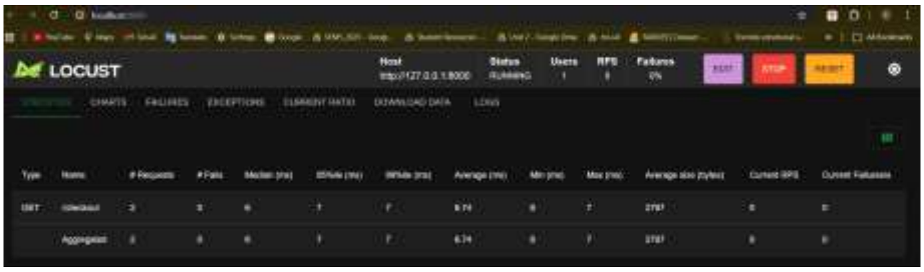
What you should observe

- 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.

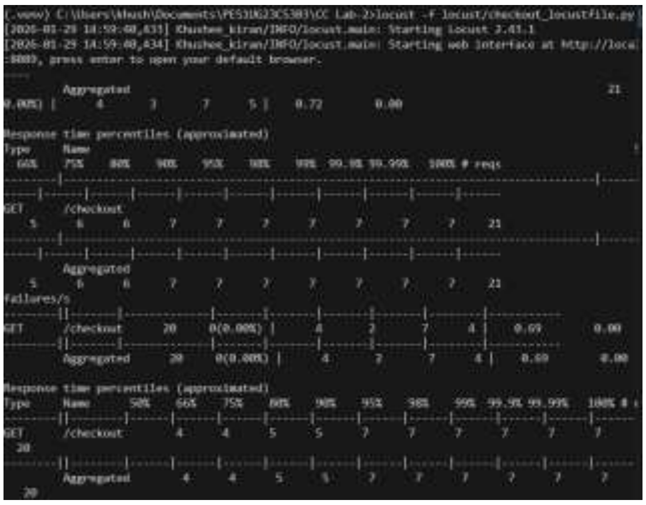
Next Lab: Split this monolith into Microservices (Events / Registration / Checkout).

CC Week 3 • Monolithic Applications Lab

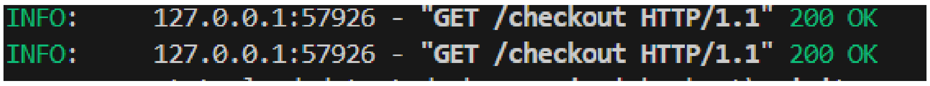
PART 5: Load Testing using Locust



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	/checkout	3	0	6	7	7	6.74	6	7	2787	0	0
	Aggregated	3	0	6	7	7	6.74	6	7	2787	0	0

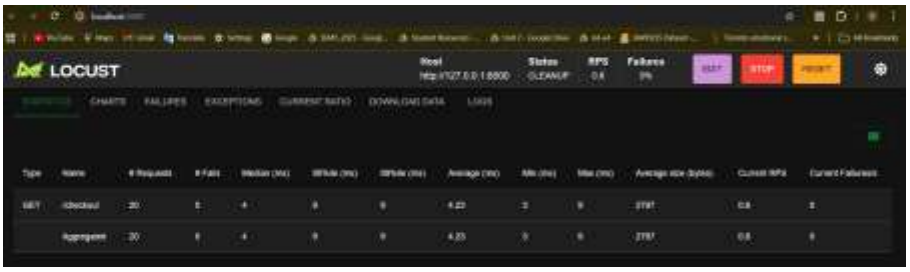


```
(.venv) C:\Users\khush\Documents\PES1UG23CS303\CC Lab-2>locust -f locust/checkout_locustfile.py
[2026-01-29 14:59:40,433] Khushee_kiran/INFO/locust.main: Starting Locust 2.43.1
[2026-01-29 14:59:40,434] Khushee_kiran/INFO/locust.main: Starting web interface at http://loca
:8089, press enter to open your default browser.
-----
         Aggregated                                                                    21
0.00%) |      4       3       7       5 |    0.72        0.00

Response time percentiles (approximated)
Type     Name
  66%     75%     80%     90%     95%     98%     99%  99.9% 99.99%    100% # reqs
--------|-------------------------------------------------------------------------|------
------|-------|-------|-------|-------|-------|-------|-------|-------|-------|------
GET      /checkout
     5      6      6      7      7      7      7      7      7      7     21
--------|-------------------------------------------------------------------------|------
------|-------|-------|-------|-------|-------|-------|-------|-------|-------|------
         Aggregated
     5      6      6      7      7      7      7      7      7      7     21
Failures/s
--------||--------|--------------|--------|--------|--------|--------|--------------
GET      /checkout      20     0(0.00%) |      4       2       7       4 |    0.69       0.00
--------||--------|--------------|--------|--------|--------|--------|--------------
         Aggregated      20     0(0.00%) |      4       2       7       4 |    0.69       0.00

Response time percentiles (approximated)
Type     Name      50%    66%    75%    80%    90%    95%    98%    99%  99.9% 99.99%   100% # r
--------||--------|------|------|------|------|------|------|------|------|------|------|------
GET      /checkout       4      4      5      5      7      7      7      7      7      7
  20
--------||--------|------|------|------|------|------|------|------|------|------|------|------
         Aggregated       4      4      5      5      7      7      7      7      7      7
   20
```

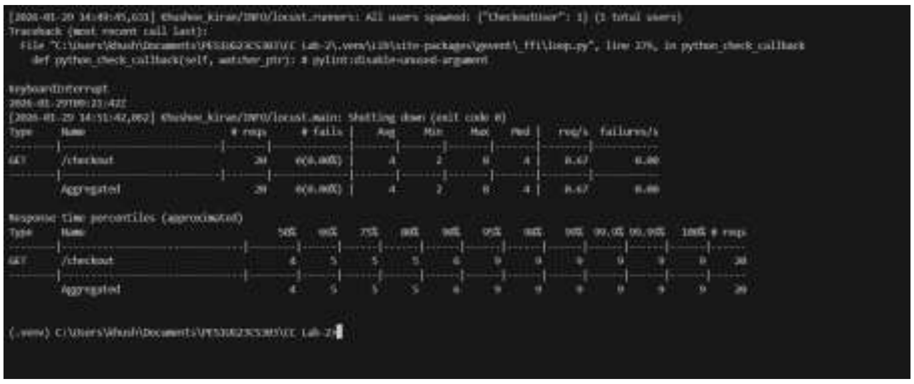


```
INFO:     127.0.0.1:57926 - "GET /checkout HTTP/1.1" 200 OK
INFO:     127.0.0.1:57926 - "GET /checkout HTTP/1.1" 200 OK
```

PART 6: Optimize the Checkout Route



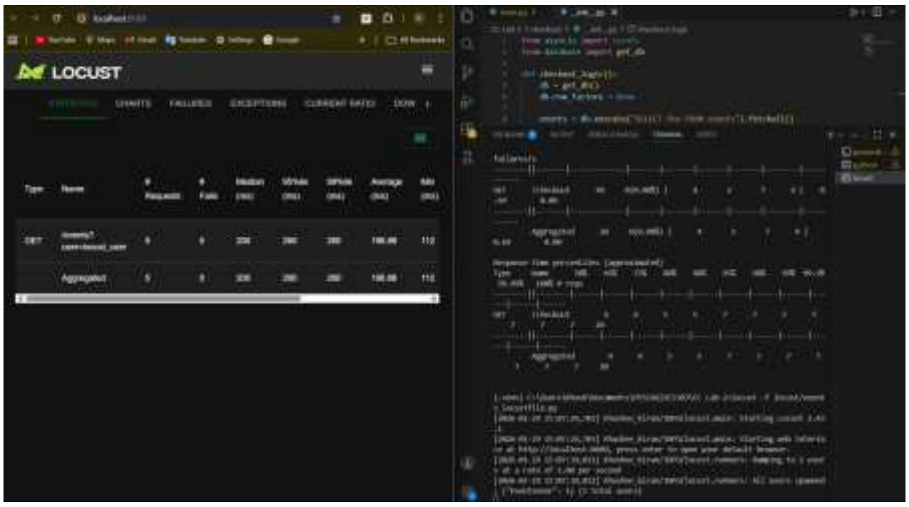
Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/checkout	20	0	4	9	9	4.23	2	9	2787	0.8	0
	Aggregated	20	0	4	9	9	4.23	2	9	2787	0.8	0



```
[2026-01-29 14:49:45,631] Khushee_kiran/INFO/locust.runners: All users spawned: {"CheckoutUser": 1} (1 total users)
Traceback (most recent call last):
  File "C:\Users\khush\Documents\PES1UG23CS303\CC Lab-2\.venv\lib\site-packages\gevent\_ffi\loop.py", line 270, in python_check_callback
    def python_check_callback(self, watcher_ptr): # pylint:disable=unused-argument

KeyboardInterrupt
2026-01-29T09:21:42Z
[2026-01-29 14:51:42,062] Khushee_kiran/INFO/locust.main: Shutting down (exit code 0)
Type     Name                                  # reqs      # fails |    Avg     Min     Max     Med |   req/s  failures/s
--------|------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
GET      /checkout                                 20     0(0.00%) |      4       2       8       4 |    0.67        0.00
--------|------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
         Aggregated                                20     0(0.00%) |      4       2       8       4 |    0.67        0.00

Response time percentiles (approximated)
Type     Name                                          50%    66%    75%    80%    90%    95%    98%    99%  99.9% 99.99%   100% # reqs
--------|--------------------------------------------|------|------|------|------|------|------|------|------|------|------|------|------
GET      /checkout                                       4      5      5      5      6      9      9      9      9      9      9     20
--------|--------------------------------------------|------|------|------|------|------|------|------|------|------|------|------|------
         Aggregated                                      4      5      5      5      6      9      9      9      9      9      9     20

(.venv) C:\Users\khush\Documents\PES1UG23CS303\CC Lab-2>
```

PART 7: Optimise *events* and *my_events*(DIY)

Events:

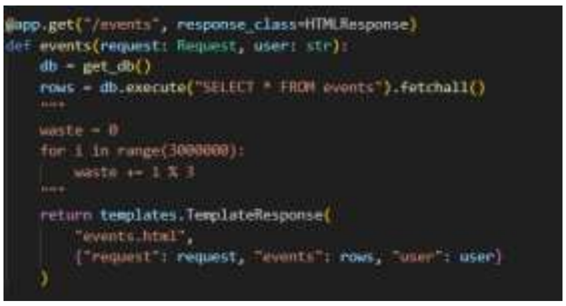


Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)
GET	/events?user=locust_user	5	0	200	290	290	198.88	112
	Aggregated	5	0	200	290	290	198.88	112

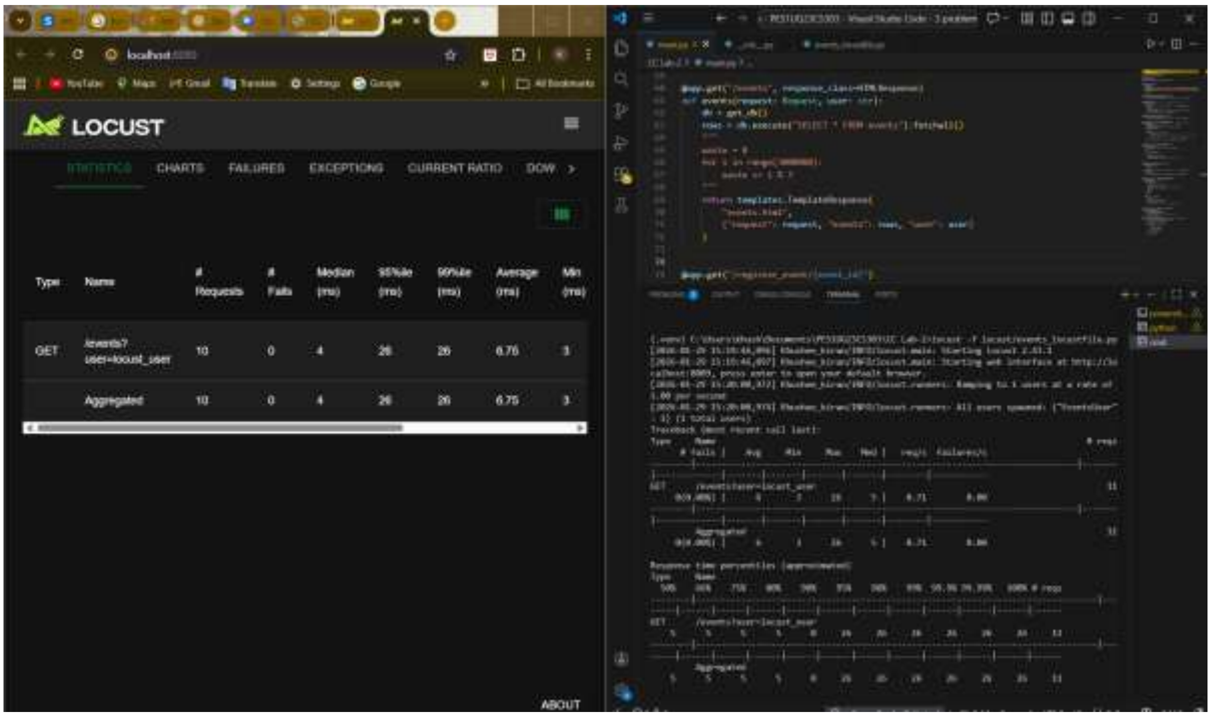
```
from fastapi import Depends
from database import get_db

def checkout_logic():
    db = get_db()
    db.row_factory = None

    events = db.execute("SELECT * FROM events").fetchall()
```

```
[illegible]
```

After optimization:



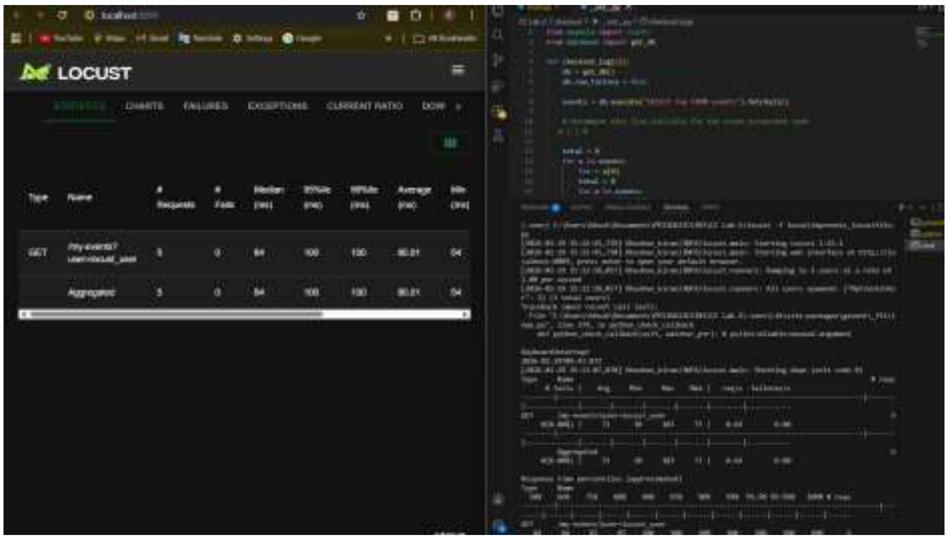
```
@app.get("/events", response_class=HTMLResponse)
def events(request: Request, user: str):
    db = get_db()
    rows = db.execute("SELECT * FROM events").fetchall()
    """
    waste = 0
    for i in range(3000000):
        waste += i % 3
    """
    return templates.TemplateResponse(
        "events.html",
        {"request": request, "events": rows, "user": user}
    )
```



Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)
GET	/events?user=locust_user	10	0	4	26	26	6.75	3
	Aggregated	10	0	4	26	26	6.75	3

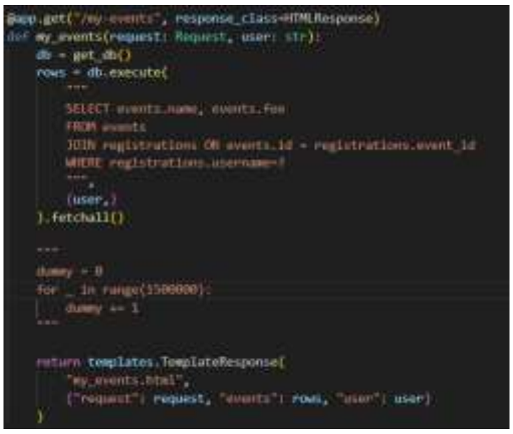
```
@app.get("/events", response_class=HTMLResponse)
def events(request: Request, user: str):
    db = get_db()
    rows = db.execute("SELECT * FROM events").fetchall()
    """
    waste = 0
    for i in range([illegible]):
        waste += i % 3
    """
    return templates.TemplateResponse(
        "events.html",
        {"request": request, "events": rows, "user": user}
    )


@app.get("/register_event/{event_id}")
```

My events:

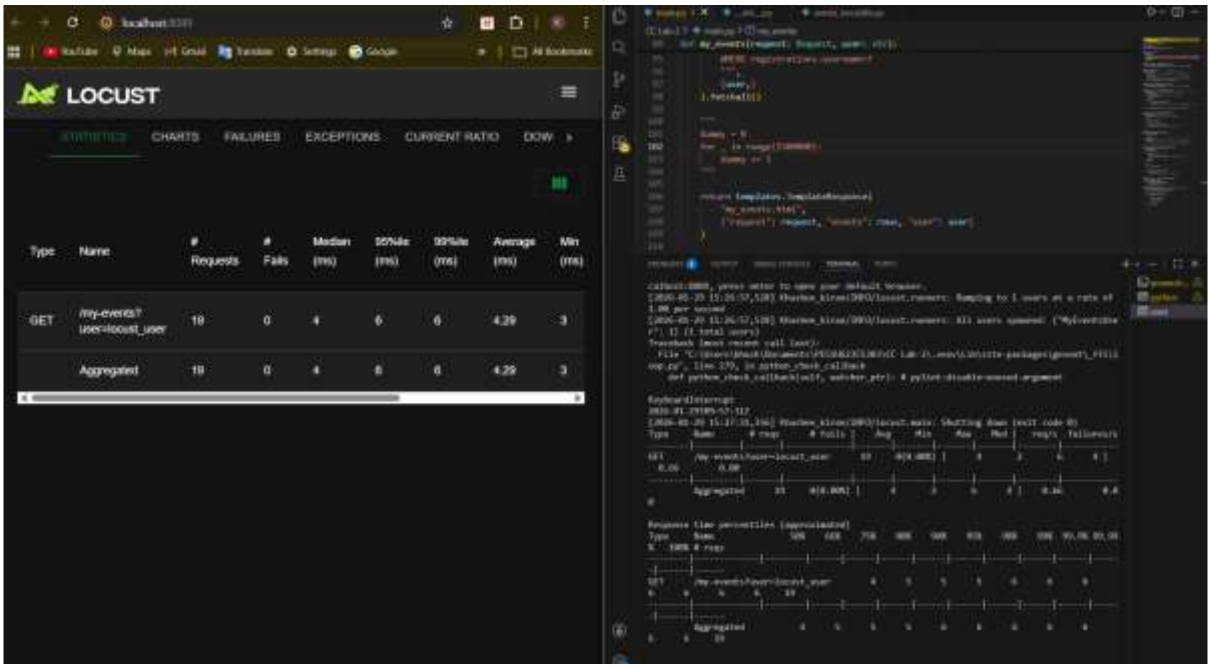


Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)
GET	/my-events?user=locust_user	5	0	84	100	130	80.81	54
	Aggregated	5	0	84	100	130	80.81	54

After optimisation:



```
@app.get("/my-events", response_class=HTMLResponse)
def my_events(request: Request, user: str):
    db = get_db()
    rows = db.execute(
        """
        SELECT events.name, events.fee
        FROM events
        JOIN registrations ON events.id = registrations.event_id
        WHERE registrations.username=?
        """,
        (user,)
    ).fetchall()

    """
    dummy = 0
    for _ in range(1500000):
        dummy += 1
    """

    return templates.TemplateResponse(
        "my_events.html",
        {"request": request, "events": rows, "user": user}
    )
```



Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)
GET	/my-events?user=locust_user	18	0	4	6	6	4.29	3
	Aggregated	18	0	4	6	6	4.29	3

```
def my_events(request: Request, user: str):
    [illegible]
        [illegible]
    ).fetchall()

    """
    dummy = 0
    for _ in range([illegible]):
        dummy += 1
    """

    return templates.TemplateResponse(
        "my_events.html",
        {"request": request, "events": rows, "user": user}
    )
```

```
[illegible]
```

Route: /events

What was the bottleneck?
The /events route had an unnecessary loop that executed around 3,000,000 times. This loop did no meaningful work and unnecessarily consumed CPU time, which slowed down the response.

What change did you make?
I removed the wasteful loop and directly returned the event data fetched from the database to the template.

Why did the performance improve?
By removing unnecessary computations, the server had to perform fewer operations, which reduced the response time and improved overall performance.

Route: /my-events

What was the bottleneck?
The /my-events route also contained an unnecessary loop running for millions of iterations, leading to high CPU usage and slower responses.

What change did you make?
I removed the redundant loop and directly returned the events registered by the user from the database.

Why did the performance improve?
Eliminating the extra loop reduced processing overhead, allowing the endpoint to respond faster and more efficiently.