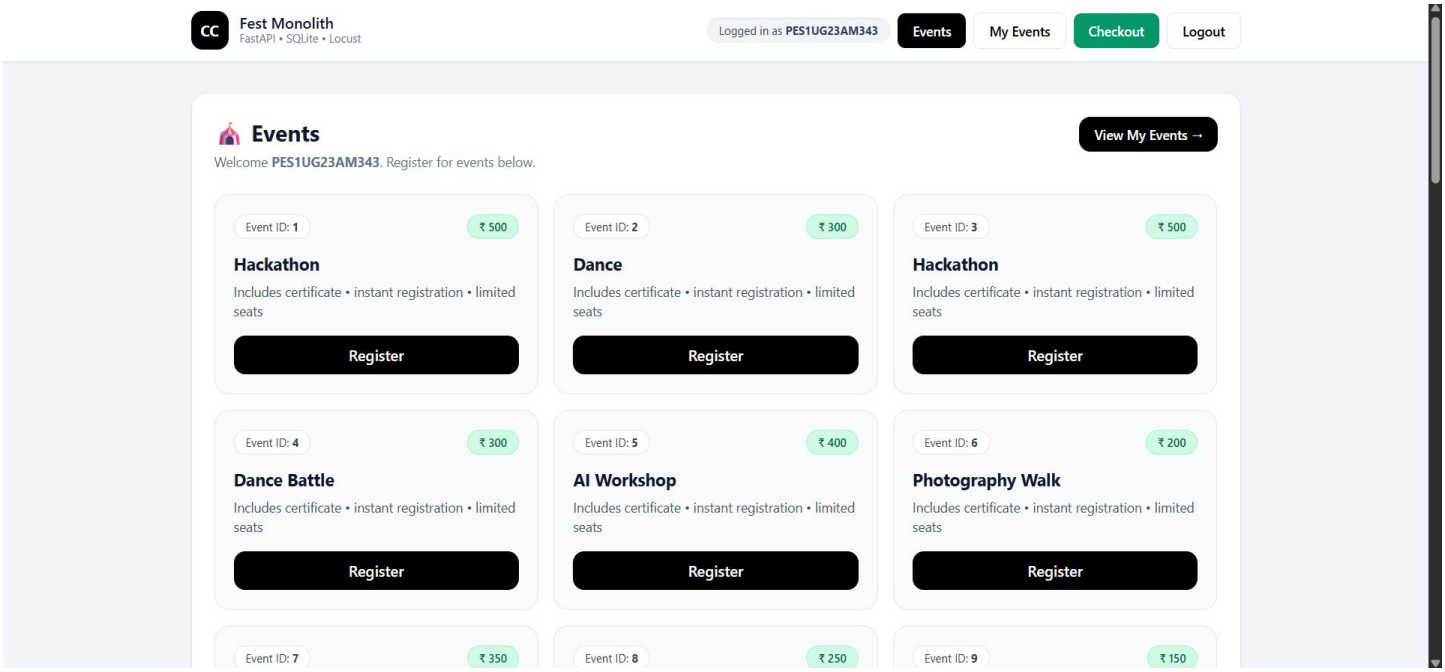
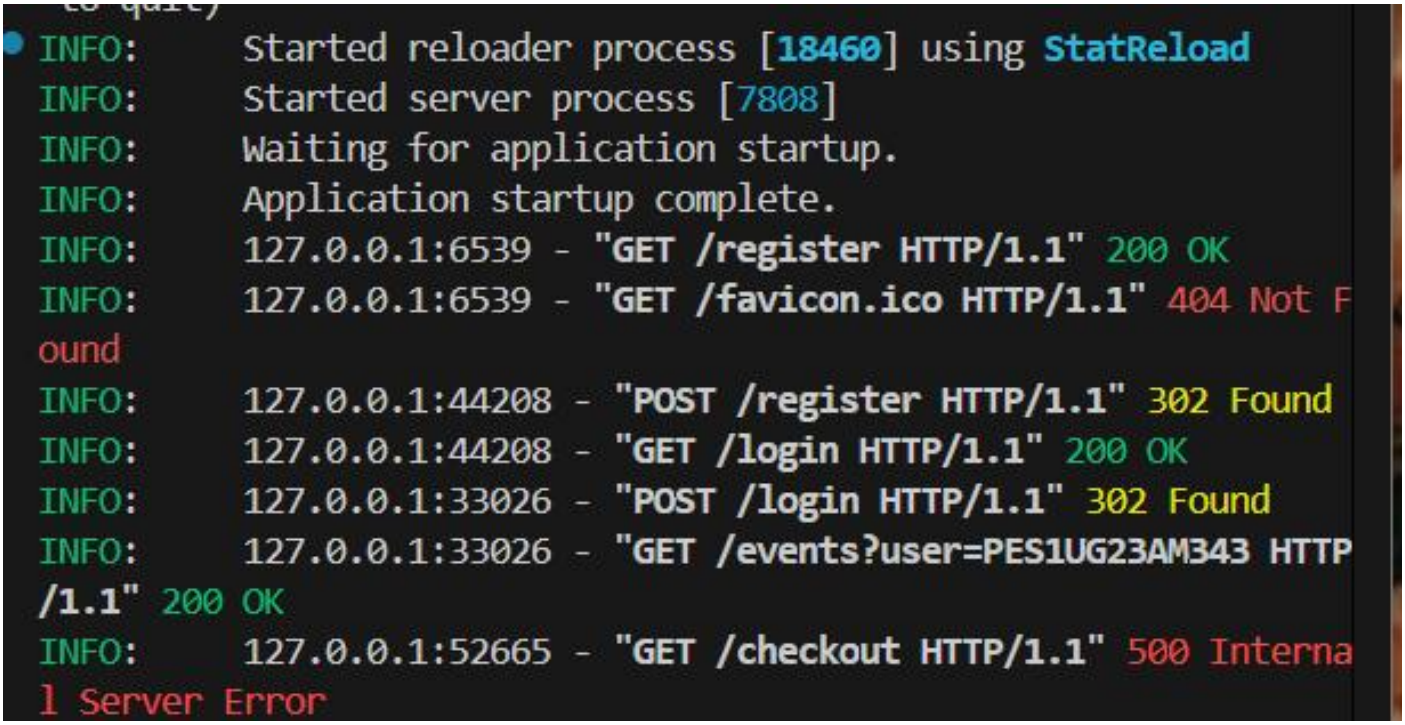
Cloud Computing

LAB-2

Name: Vardha Kathuria
Section: F
SRN:PES1UG23AM343


CC
Fest Monolith
FastAPI • SQLite • Locust
Logged in as PES1UG23AM343
Events
My Events
Checkout
Logout
Events
View My Events →
Welcome PES1UG23AM343. Register for events below.
Event ID: 1
₹ 500
Hackathon
Includes certificate • instant registration • limited seats
Register
Event ID: 2
₹ 300
Dance
Includes certificate • instant registration • limited seats
Register
Event ID: 3
₹ 500
Hackathon
Includes certificate • instant registration • limited seats
Register
Event ID: 4
₹ 300
Dance Battle
Includes certificate • instant registration • limited seats
Register
Event ID: 5
₹ 400
AI Workshop
Includes certificate • instant registration • limited seats
Register
Event ID: 6
₹ 200
Photography Walk
Includes certificate • instant registration • limited seats
Register
Event ID: 7
₹ 350
Event ID: 8
₹ 250
Event ID: 9
₹ 150


INFO: Started reloader process [18460] using StatReload
INFO: Started server process [7808]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: 127.0.0.1:6539 - "GET /register HTTP/1.1" 200 OK
INFO: 127.0.0.1:6539 - "GET /favicon.ico HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:44208 - "POST /register HTTP/1.1" 302 Found
INFO: 127.0.0.1:44208 - "GET /login HTTP/1.1" 200 OK
INFO: 127.0.0.1:33026 - "POST /login HTTP/1.1" 302 Found
INFO: 127.0.0.1:33026 - "GET /events?user=PES1UG23AM343 HTTP/1.1" 200 OK
INFO: 127.0.0.1:52665 - "GET /checkout HTTP/1.1" 500 Internal Server Error

Route 1: checkout

CC Fest Monolith
FastAPI • SQLite • Locust

Login | Create Account

Monolith Failure

HTTP 500

One bug in one module impacted the **entire application**.

Error Message
`division by zero`

Why did this happen?

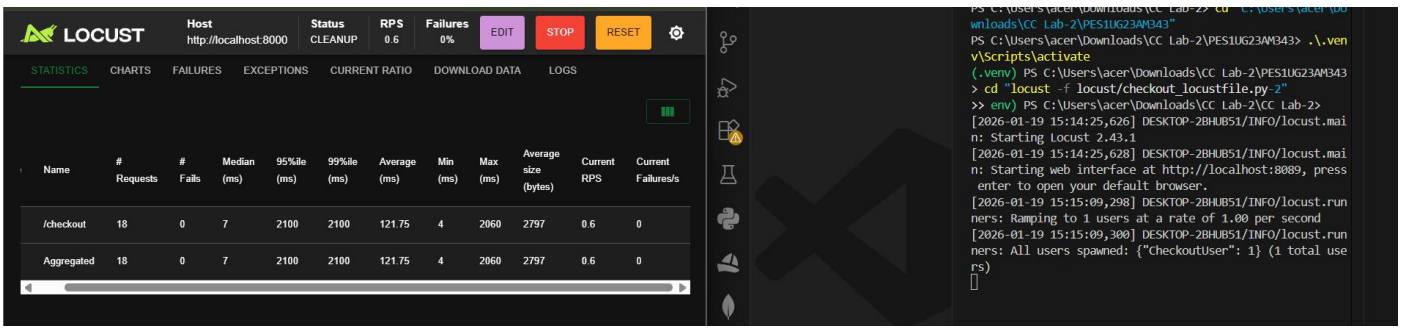
Because this is a **monolithic application**: all modules share the same runtime and deployment. When one feature crashes, it affects the whole system.

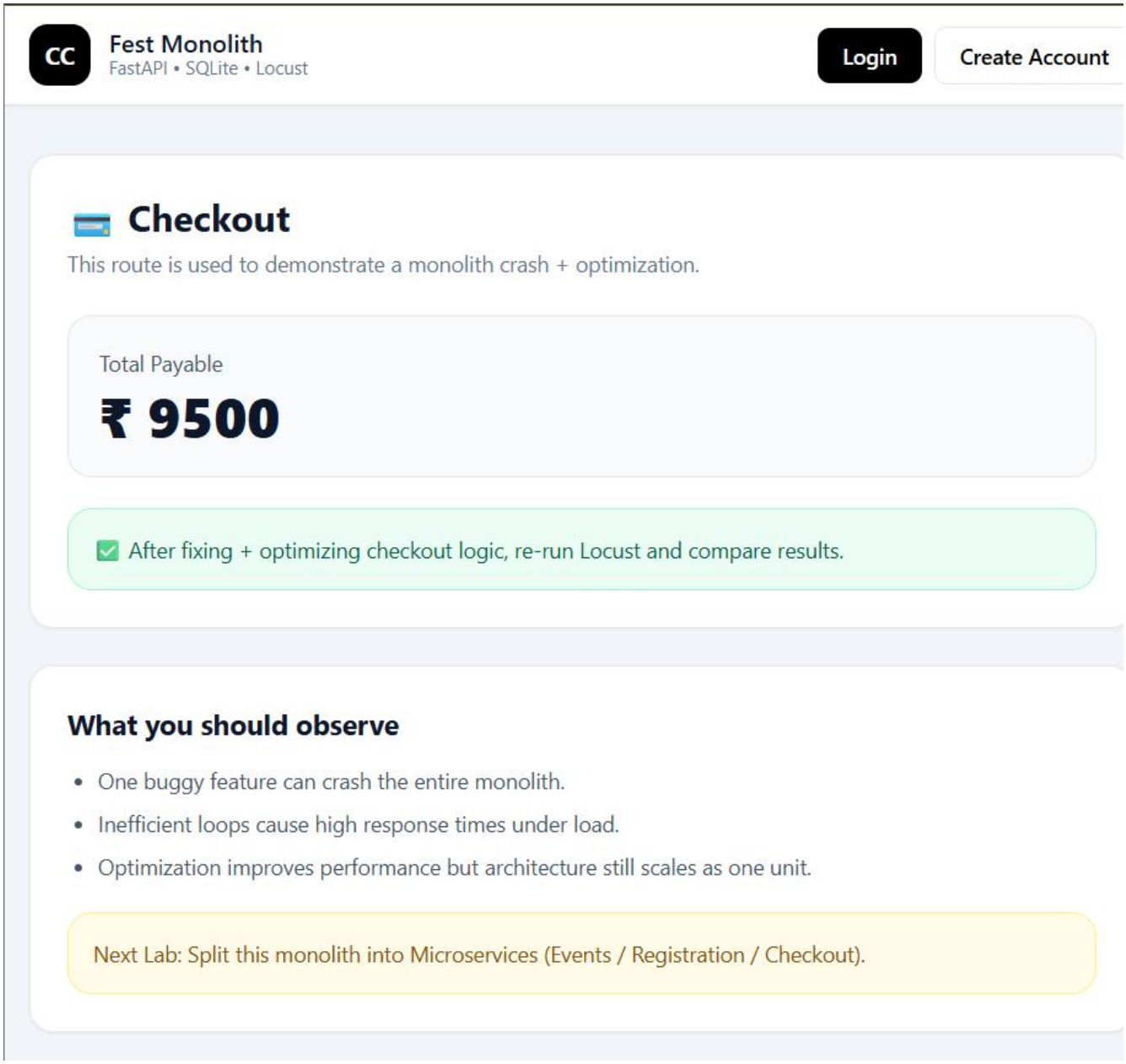
What should you do in the lab?

- Take a screenshot (crash demonstration)
- Fix the bug in the indicated module
- Restart the server and verify recovery

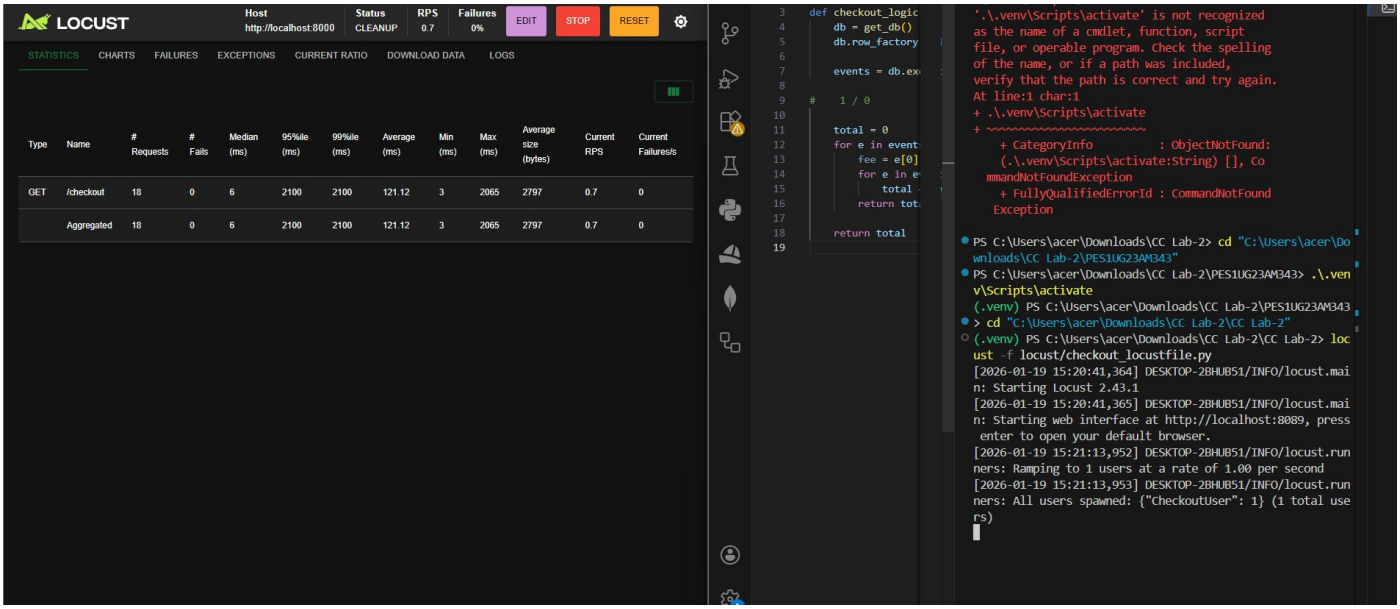
Back to Events | Login

Before optimization:





After optimization:



Bottleneck

The /checkout route initially contained an intentional bug (division by zero) which caused a server crash. Additionally, the fee calculation logic was inefficient, as it used a loop to increment the total amount one unit at a time.

Change Made

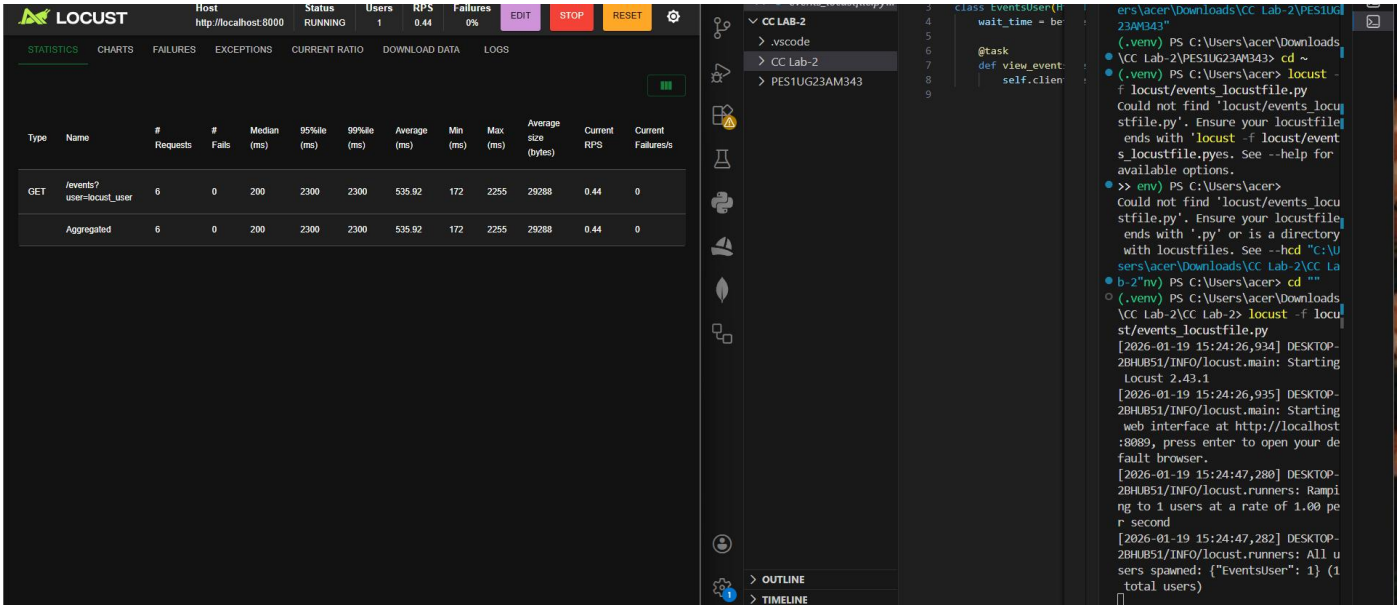
- The division-by-zero statement was commented out to prevent the application from crashing.
- The fee calculation logic was optimized by directly summing the event fees instead of using repetitive looping.

Why Performance Improved

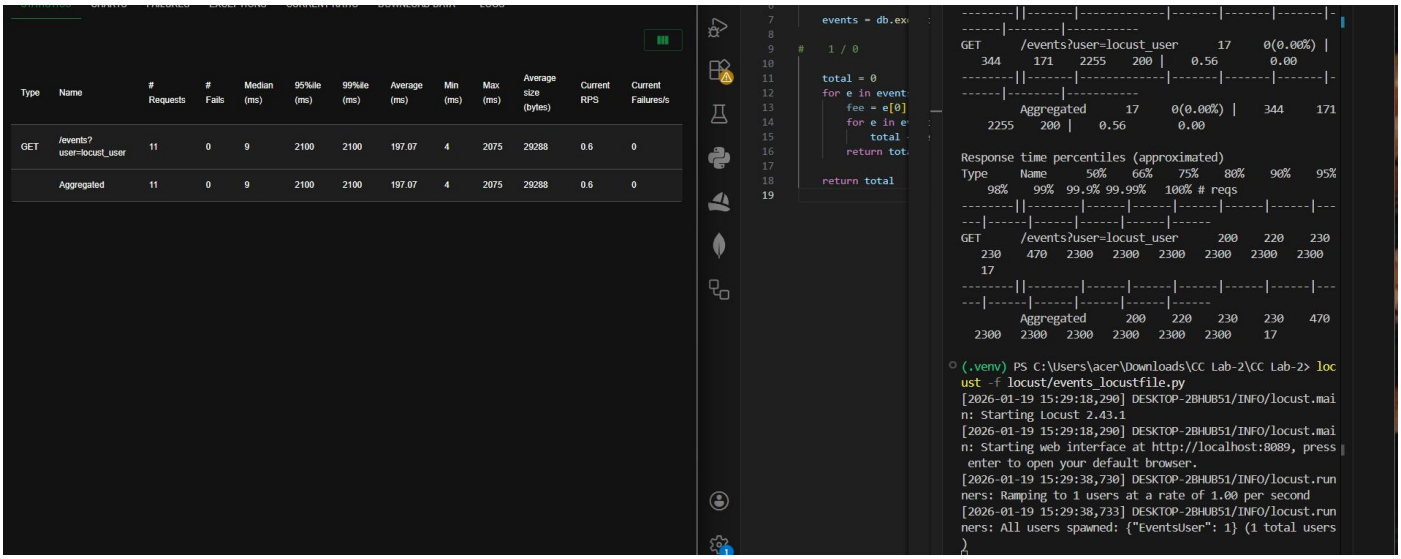
Removing the crash ensured application stability, and optimizing the fee calculation reduced unnecessary CPU computation. This resulted in a lower average response time during load testing.

Route 2: events

Before optimization:



After optimization:



Bottleneck

The /events route contained an unnecessary computation loop that executed millions of iterations without contributing to the actual functionality. This caused artificial CPU delay and increased response time for every request.

Change Made

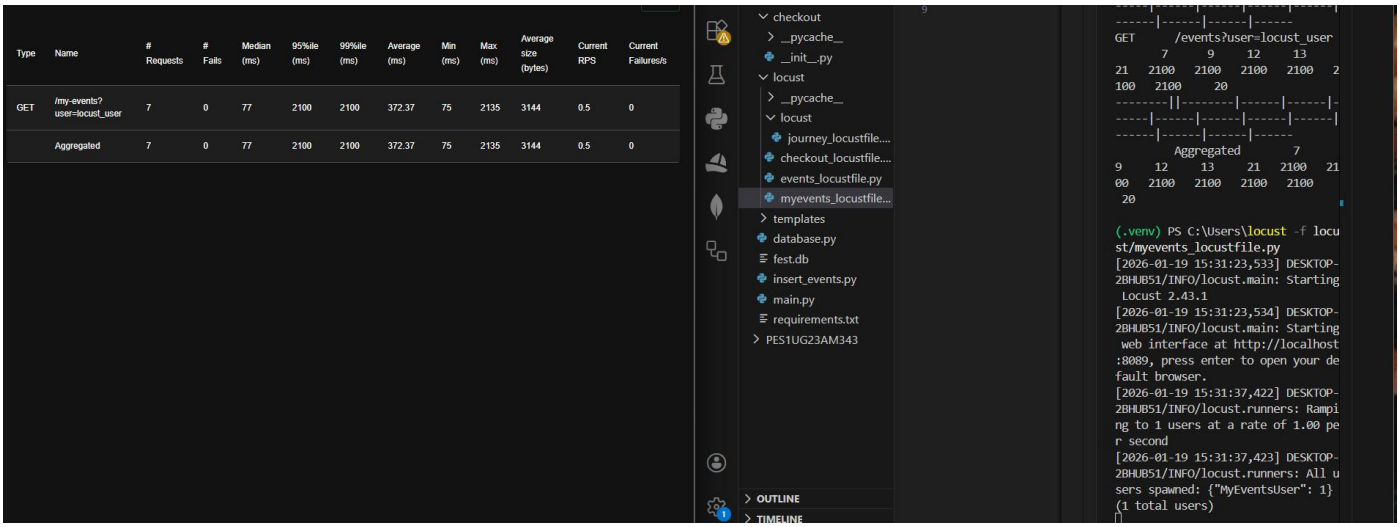
The redundant loop was completely removed, leaving only the required database query and template rendering logic.

Why Performance Improved

By eliminating unnecessary CPU-intensive operations, the server was able to respond faster to requests. This significantly reduced the average response time observed in Locust after optimization.

Route 3: myevents

Before optimization:



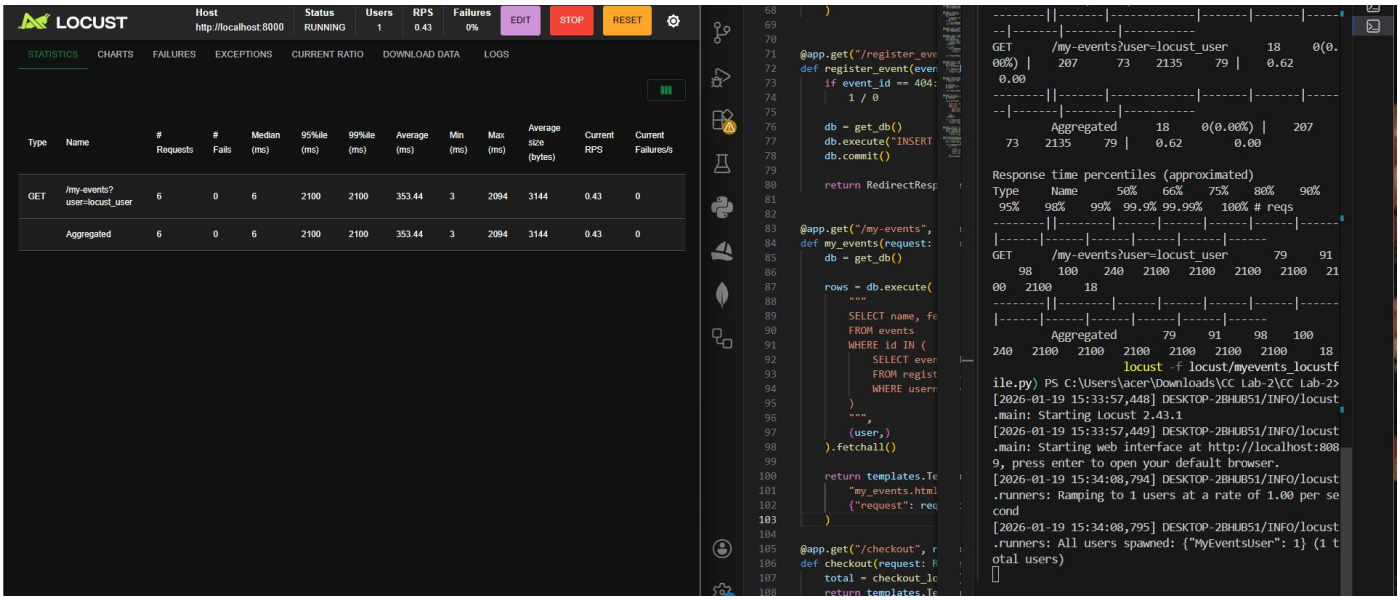
Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/my-events?user=locust_user	7	0	77	2100	2100	372.37	75	2135	3144	0.5	0
	Aggregated	7	0	77	2100	2100	372.37	75	2135	3144	0.5	0

```
checkout
  __pycache__
  __init__.py
locust
  __pycache__
  locust
    journey_locustfile....
    checkout_locustfile....
    events_locustfile.py
    myevents_locustfile...
  templates
  database.py
  fest.db
  insert_events.py
  main.py
  requirements.txt
PES1UG23AM343
OUTLINE
TIMELINE
```

```
GET      /events?user=locust_user
      7      9     12     13     21   2100   2100   2100   2100   2100   2100     20
--------||--------|------|------|------|------|------|------|------|------|------|------
        Aggregated        7      9     12     13     21   2100   2100   2100   2100   2100     20

(.venv) PS C:\Users\locust -f locust/myevents_locustfile.py
[2026-01-19 15:31:23,533] DESKTOP-2BHUB51/INFO/locust.main: Starting Locust 2.43.1
[2026-01-19 15:31:23,534] DESKTOP-2BHUB51/INFO/locust.main: Starting web interface at http://localhost:8089, press enter to open your default browser.
[2026-01-19 15:31:37,422] DESKTOP-2BHUB51/INFO/locust.runners: Ramping to 1 users at a rate of 1.00 per second
[2026-01-19 15:31:37,423] DESKTOP-2BHUB51/INFO/locust.runners: All users spawned: {"MyEventsUser": 1} (1 total users)
```

After optimization:



LOCUST — Host: http://localhost:8000 | Status: RUNNING | Users: 1 | RPS: 0.43 | 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	6	0	6	2100	2100	353.44	3	2094	3144	0.43	0
	Aggregated	6	0	6	2100	2100	353.44	3	2094	3144	0.43	0

```
)

@app.get("/register_eve
def register_event(ever
    if event_id == 404:
        1 / 0

    db = get_db()
    db.execute("INSERT
    db.commit()

    return RedirectResp


@app.get("/my-events",
def my_events(request:
    db = get_db()

    rows = db.execute(
        """
        SELECT name, fe
        FROM events
        WHERE id IN (
            SELECT ever
            FROM regist
            WHERE userr
        )
        """
        ,
        (user,)
    ).fetchall()

    return templates.Te
        "my_events.html
        {"request": req
    )

@app.get("/checkout", r
def checkout(request: R
    total = checkout_lc
    return templates.Te
```

```
--------||-------|-------------|-------|-------|-------|-------|-------|-------|-----------
GET      /my-events?user=locust_user      18     0(0.00%) |    207      73    2135     79 |    0.62         0.00
--------||-------|-------------|-------|-------|-------|-------|-------|-------|-----------
        Aggregated      18     0(0.00%) |    207      73    2135     79 |    0.62         0.00

Response time percentiles (approximated)
Type     Name      50%    66%    75%    80%    90%    95%    98%    99%  99.9% 99.99%   100% # reqs
--------||--------|------|------|------|------|------|------|------|------|------|------|------
GET      /my-events?user=locust_user       79     91     98    100    240   2100   2100   2100   2100   2100   2100     18
--------||--------|------|------|------|------|------|------|------|------|------|------|------
        Aggregated       79     91     98    100    240   2100   2100   2100   2100   2100   2100     18

                    locust -f locust/myevents_locustfile.py) PS C:\Users\acer\Downloads\CC Lab-2\CC Lab-2>
[2026-01-19 15:33:57,448] DESKTOP-2BHUB51/INFO/locust.main: Starting Locust 2.43.1
[2026-01-19 15:33:57,449] DESKTOP-2BHUB51/INFO/locust.main: Starting web interface at http://localhost:8089, press enter to open your default browser.
[2026-01-19 15:34:08,794] DESKTOP-2BHUB51/INFO/locust.runners: Ramping to 1 users at a rate of 1.00 per second
[2026-01-19 15:34:08,795] DESKTOP-2BHUB51/INFO/locust.runners: All users spawned: {"MyEventsUser": 1} (1 total users)
```

Bottleneck

The /my-events route had an artificial delay introduced through a large loop performing over a million iterations. Additionally, the original JOIN-based query added extra processing overhead.

Changes Made

- The artificial computation loop was removed.
- The database query was simplified by using a subquery instead of a JOIN to fetch only the required event details.

Why Performance Improved

Removing the redundant computation eliminated CPU wastage, and simplifying the database query reduced execution overhead. Together, these changes improved response time and overall performance under load.

Conclusion

This lab demonstrated the limitations of monolithic architecture, where inefficiencies or errors in a single module can impact the entire application. By identifying bottlenecks and optimizing individual routes, measurable performance improvements were observed using Locust. This highlights the importance of efficient backend logic and serves as a motivation for adopting more modular architectures such as microservices.