CC LAB 2

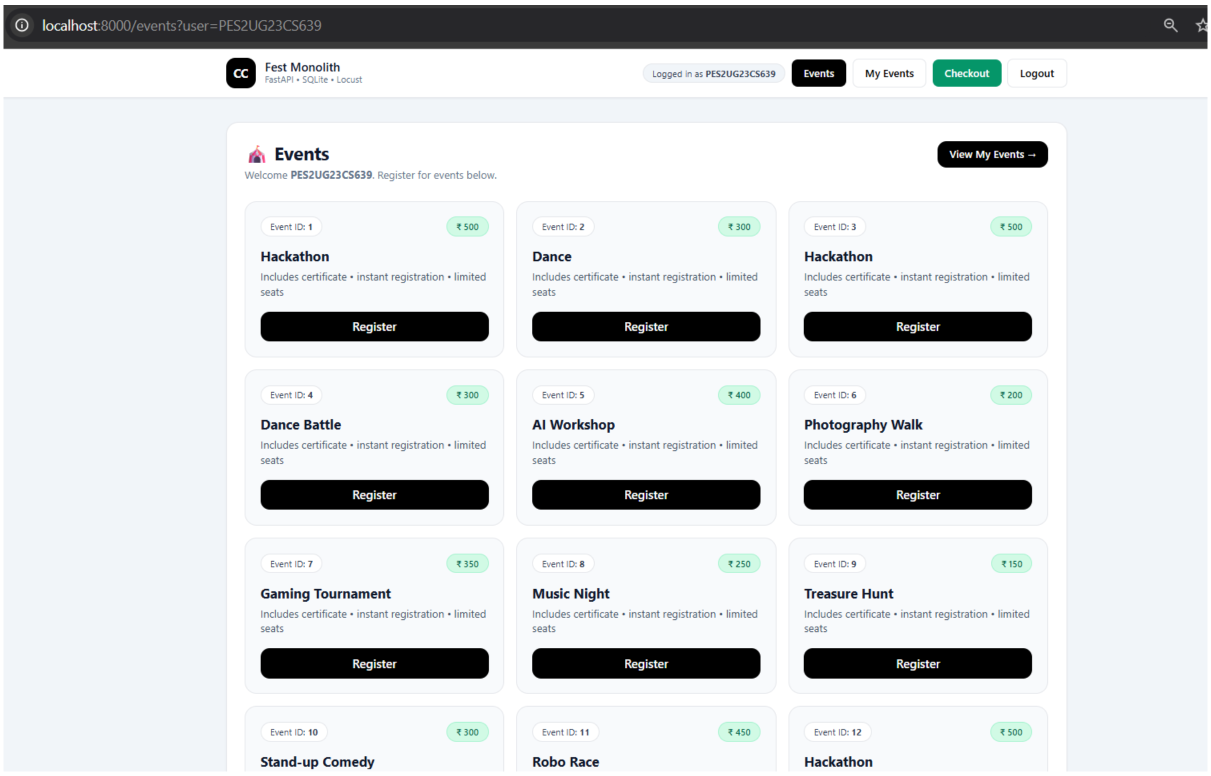
NAME- TAHIR SHAFIQ

SRN- PES2UG23CS639

SEC J

GITHUB LINK- https://github.com/tahirshafiq398/Monolithic

Ss1



localhost:8000/events?user=PES2UG23CS639

CC Fest Monolith
FastAPI • SQLite • Locust

Logged in as PES2UG23CS639 | Events | My Events | Checkout | Logout

Events

Welcome **PES2UG23CS639**. Register for events below.

View My Events →

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
Gaming Tournament
Includes certificate • instant registration • limited seats
Register

Event ID: 8 — ₹ 250
Music Night
Includes certificate • instant registration • limited seats
Register

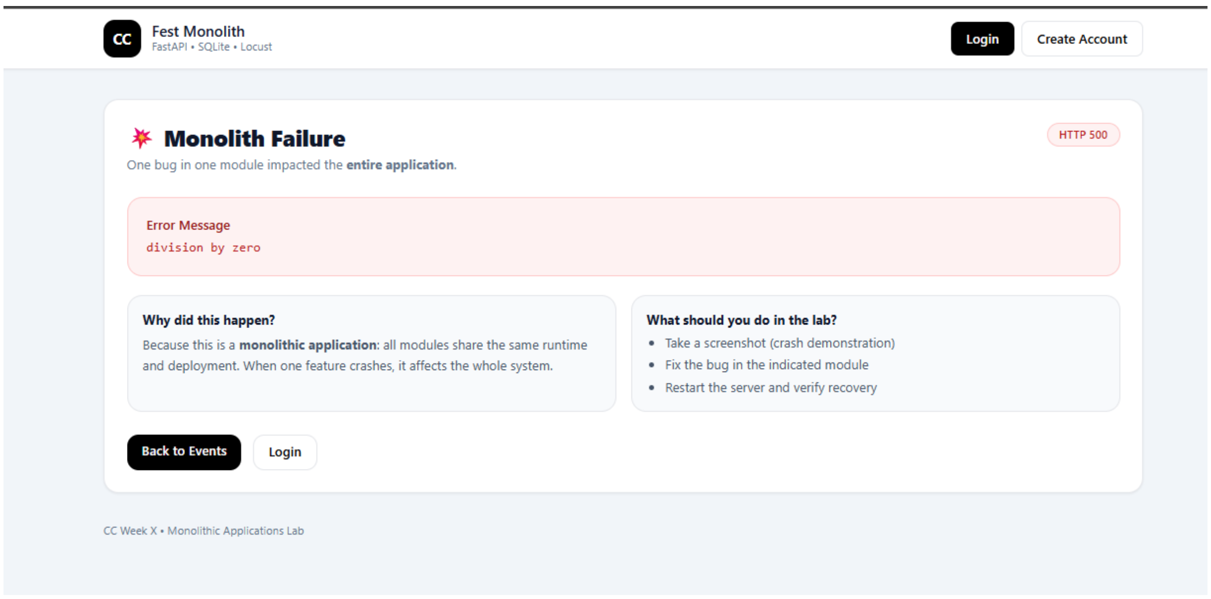
Event ID: 9 — ₹ 150
Treasure Hunt
Includes certificate • instant registration • limited seats
Register

Event ID: 10 — ₹ 300
Stand-up Comedy

Event ID: 11 — ₹ 450
Robo Race

Event ID: 12 — ₹ 500
Hackathon

ss2



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.

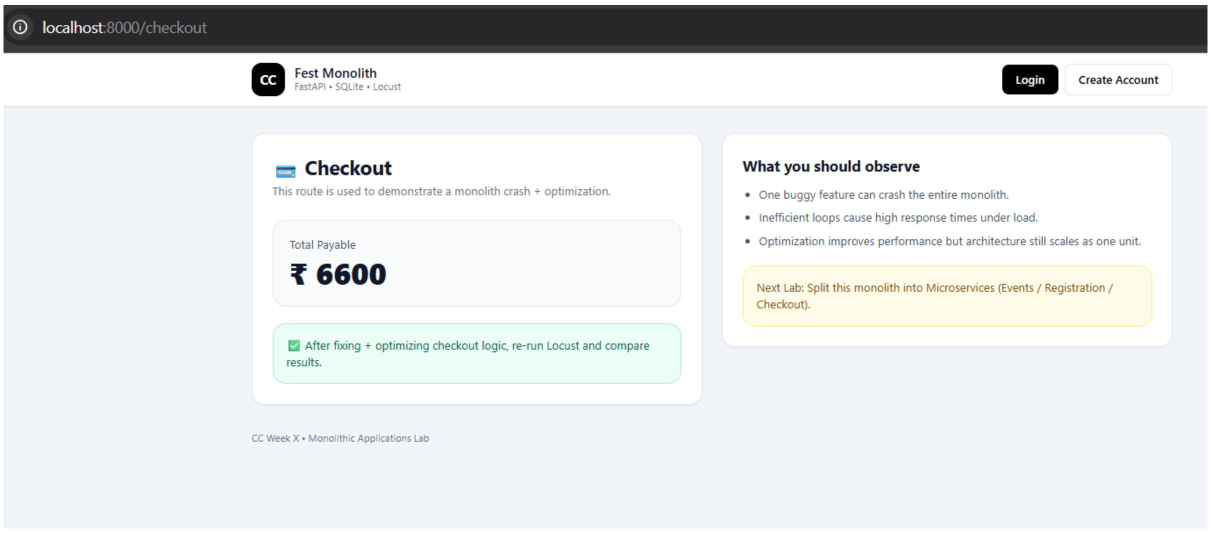
What should you do in the lab?

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

Back to Events | Login

CC Week X • Monolithic Applications Lab

Ss3



localhost:8000/checkout

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.

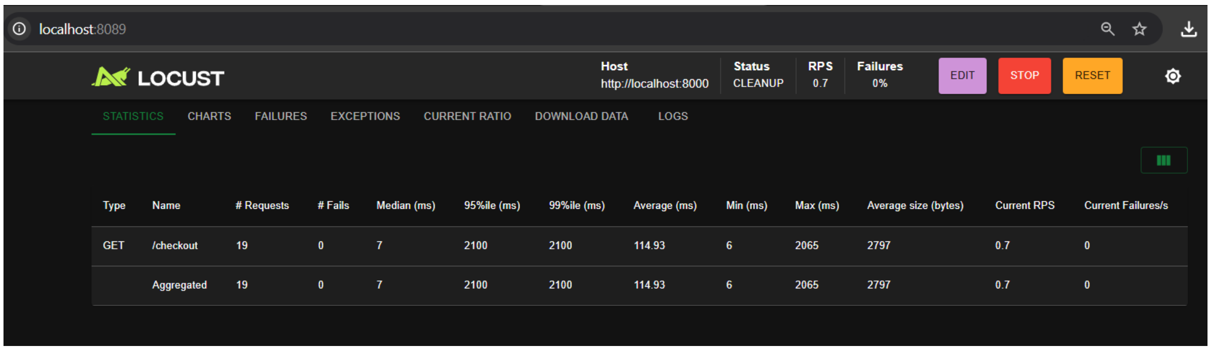
What you should observe

- 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 X • Monolithic Applications Lab

Ss4



localhost:8089

LOCUST

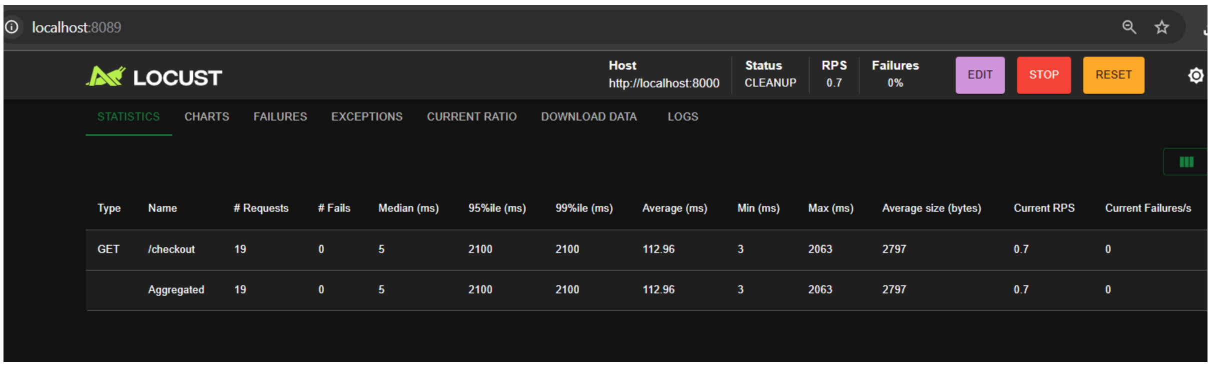
Host: http://localhost:8000 | Status: CLEANUP | RPS: 0.7 | Failures: 0%

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	/checkout	19	0	7	2100	2100	114.93	6	2065	2797	0.7	0
	Aggregated	19	0	7	2100	2100	114.93	6	2065	2797	0.7	0

Ss5



localhost:8089

LOCUST

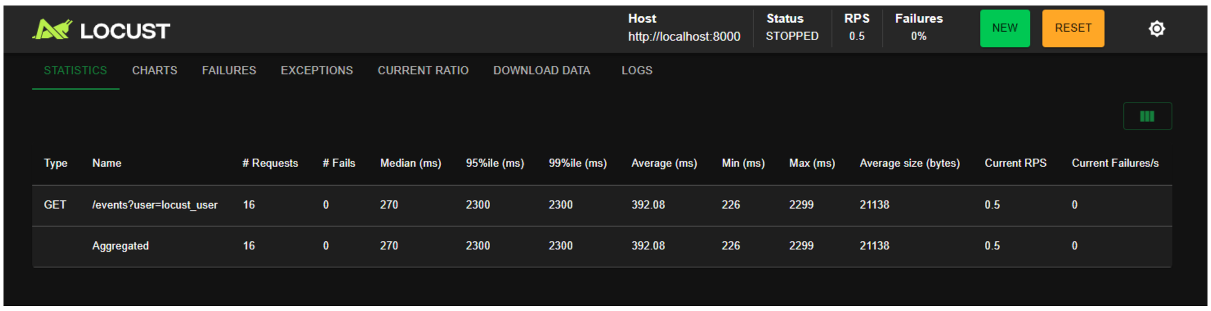
Host: http://localhost:8000 | Status: CLEANUP | RPS: 0.7 | Failures: 0%

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	/checkout	19	0	5	2100	2100	112.96	3	2063	2797	0.7	0
	Aggregated	19	0	5	2100	2100	112.96	3	2063	2797	0.7	0

Ss6



LOCUST

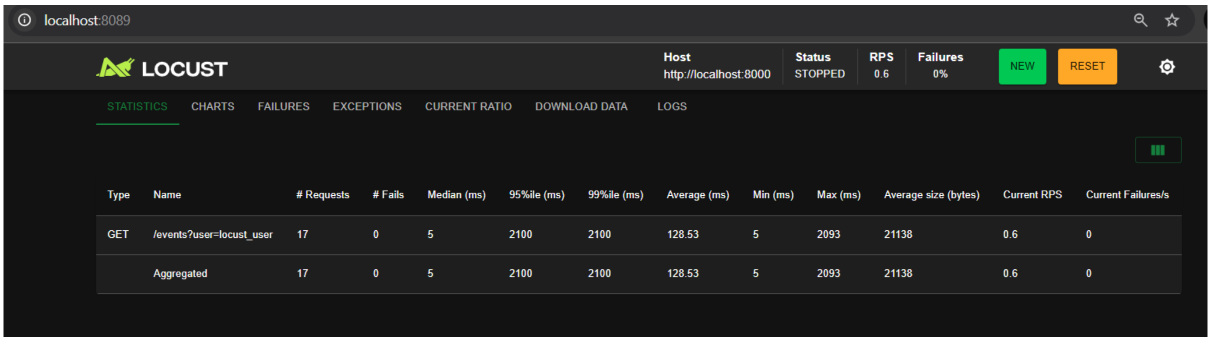
Host: http://localhost:8000 | Status: STOPPED | RPS: 0.5 | Failures: 0%

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	16	0	270	2300	2300	392.08	226	2299	21138	0.5	0
	Aggregated	16	0	270	2300	2300	392.08	226	2299	21138	0.5	0

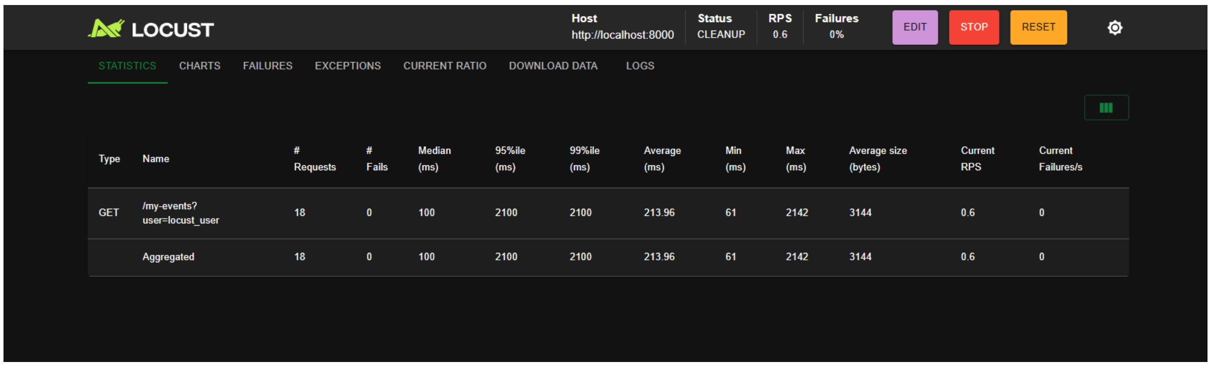
Ss7



Host http://localhost:8000 | Status STOPPED | RPS 0.6 | Failures 0%

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	17	0	5	2100	2100	128.53	5	2093	21138	0.6	0
	Aggregated	17	0	5	2100	2100	128.53	5	2093	21138	0.6	0

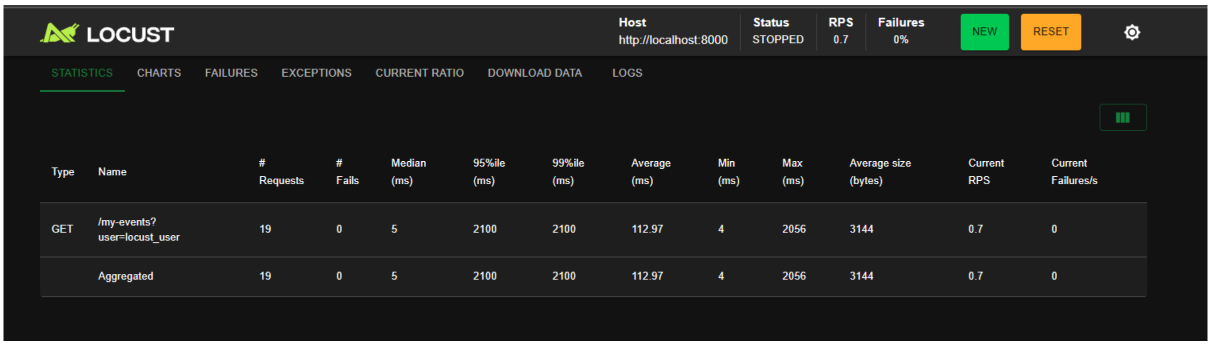
Ss8



Host http://localhost:8000 | Status CLEANUP | RPS 0.6 | Failures 0%

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	100	2100	2100	213.96	61	2142	3144	0.6	0
	Aggregated	18	0	100	2100	2100	213.96	61	2142	3144	0.6	0

Ss9



Host http://localhost:8000 | Status STOPPED | RPS 0.7 | Failures 0%

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	19	0	5	2100	2100	112.97	4	2056	3144	0.7	0
	Aggregated	19	0	5	2100	2100	112.97	4	2056	3144	0.7	0

1. /events Route

What was the bottleneck?

The route contained an unnecessary computation loop that ran for millions of iterations, even though it did not contribute to generating the response. This caused high CPU usage and increased response time.

What change did you make?

The redundant loop was removed so that the route only performs the required database query and template rendering.

Why did the performance improve?

By eliminating unnecessary CPU-intensive operations, the server processes requests faster, resulting in reduced response time and better throughput.

2. /my-events Route

What was the bottleneck?
An artificial delay loop was introduced that performed a large number of iterations without doing useful work, increasing the response time of the route.

What change did you make?
The delay loop was removed, allowing the route to directly return the database query results.

Why did the performance improve?
Removing the delay reduced CPU overhead and allowed faster request handling, leading to significantly improved response time.