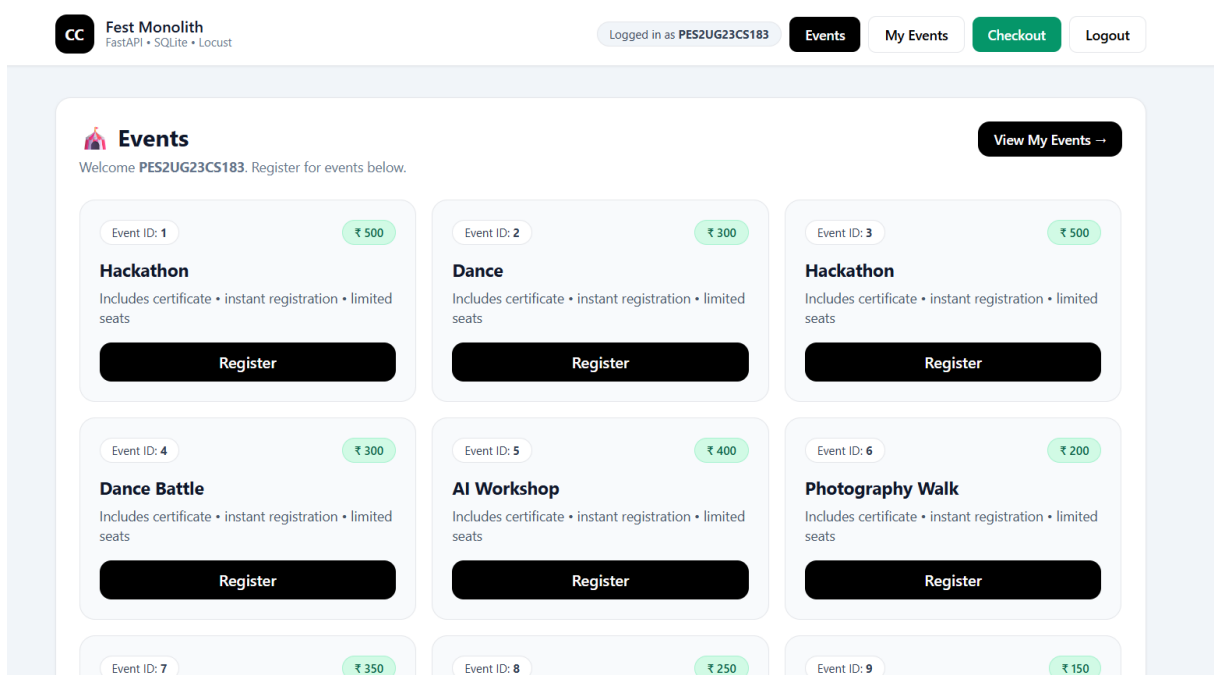


CLOUD COMPUTING – LAB 2 – MONOLITHIC ARCHITECTURE

Name	Diya D Bhat
SRN	PES2UG23CS183
Section	C
Date	20/01/26

PART 2: Use the Application

Screenshot 1 (SS1)



PART 3: Observe Monolithic Failure (Crash)

Screenshot 2 (SS2)

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

```
(.venv) (base) PS C:\Users\diyab\sem6\CC\PES2UG23CS183\CC_LAB2> uvicorn main:app --reload
INFO:      Application startup complete.
INFO:      127.0.0.1:53023 - "GET /checkout HTTP/1.1" 500 Internal Server Error
ERROR:      Exception in ASGI application
```

PART 4: Fix the Bug

Screenshot 3 (SS3)

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

```
INFO:      Application startup complete.
INFO:      127.0.0.1:61439 - "GET /checkout HTTP/1.1" 200 OK
```

PART 5: Load Testing using Locust

Screenshot 4 (SS4)

 **LOCUST**

Host

http://localhost:8000

Status

STOPPED

RPS


0.7

Failures

0%

NEW

RESET



STATISTICSCHARTSFAILURES EXCEPTIONSCURRENT RATIODOWNLOAD DATALOGS

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	113.27	4	2066	2797	0.7	0
	Aggregated	19	0	5	2100	2100	113.27	4	2066	2797	0.7	0


```
[2026-01-20 14:44:44,285] lenovo_diya/INFO/locust.runners: Ramping to 1 users at a rate of 1.00 per second
[2026-01-20 14:44:44,286] lenovo_diya/INFO/locust.runners: All users spawned: {"CheckoutUser": 1} (1 total users)
Traceback (most recent call last):
[2026-01-20 14:45:54,063] lenovo_diya/INFO/locust.main: Shutting down (exit code 0)
Type      Name      # reqs      # fails      Avg      Min      Max      Med      req/s      failures/s
-----
GET      /checkout      18      0(0.00%)      117      3      2030      5      0.64      0.00
-----
[2026-01-20 14:45:54,063] lenovo_diya/INFO/locust.main: Shutting down (exit code 0)
Type      Name      # reqs      # fails      Avg      Min      Max      Med      req/s      failures/s
-----
GET      /checkout      18      0(0.00%)      117      3      2030      5      0.64      0.00
-----
Aggregated      18      0(0.00%)      117      3      2030      5      0.64      0.00
-----

Response time percentiles (approximated)
Type      Name      50%      66%      75%      80%      90%      95%      98%      99%      99.9%      99.99%      100%      # reqs
-----
GET      /checkout      5      6      6      7      7      2000      2000      2000      2000      2000      2000      18
-----
Aggregated      5      6      6      7      7      2000      2000      2000      2000      2000      2000      18
-----

(.venv) (base) PS C:\Users\diya\sem6\CC\PES2UG23CS183\CC_LAB2>
```

PART 6: Optimize the Checkout Route

Screenshot 5 (SS5)

 **LOCUST**

Host

http://localhost:8000

Status

RUNNING

Users

0

RPS

0.7


Failures

0%

EDIT

STOP

RESET



STATISTICSCHARTSFAILURES EXCEPTIONSCURRENT RATIODOWNLOAD DATALOGS

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	5	2000	2000	106.81	4	2032	2797	0.7	0
	Aggregated	20	0	5	2000	2000	106.81	4	2032	2797	0.7	0


```
2026-01-20T09:22:33Z
[2026-01-20 14:52:33,535] lenovo_diya/INFO/locust.main: Shutting down (exit code 0)
Type      Name      # reqs      # fails      Avg      Min      Max      Med      req/s      failures/s
-----
GET      /checkout      19      0(0.00%)      113      3      2065      5      0.64      0.00
-----
Aggregated      19      0(0.00%)      113      3      2065      5      0.64      0.00
-----

Response time percentiles (approximated)
Type      Name      50%      66%      75%      80%      90%      95%      98%      99%      99.9%      99.99%      100%      # reqs
-----
GET      /checkout      5      5      5      5      6      2100      2100      2100      2100      2100      2100      19
-----
Aggregated      5      5      5      5      6      2100      2100      2100      2100      2100      2100      19
-----

(.venv) (base) PS C:\Users\diya\sem6\CC\PES2UG23CS183\CC_LAB2>
```

PART 7: Optimise events and my_events(DIY)

Screenshot 6 (SS6)

 **LOCUST**

Host
http://localhost:8000

Status
CLEANUP


RPS
0.5

Failures
0%

EDIT

STOP

RESET



STATISTICSCHARTSFAILURESEXCEPTIONSCURRENT RATIODOWNLOAD DATALOGS

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	15	0	520	2600	2600	652.79	401	2617	21138	0.5	0
	Aggregated	15	0	520	2600	2600	652.79	401	2617	21138	0.5	0

2026-01-20T09:30:05Z
[2026-01-20 15:00:05,316] lenovo_diya/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	15	0(0.00%)	652	400	2616	520	0.51	0.00
	Aggregated	15	0(0.00%)	652	400	2616	520	0.51	0.00

Response time percentiles (approximated)

Type	Name	50%	66%	75%	80%	90%	95%	98%	99%	99.9%	99.99%	100%	# reqs
GET	/events?user=locust_user	520	530	570	570	620	2600	2600	2600	2600	2600	2600	15
	Aggregated	520	530	570	570	620	2600	2600	2600	2600	2600	2600	15

(.venv) (base) PS C:\Users\diyab\sem6\CC\PES2UG23CS183\CC_LAB2>

Screenshot 7 (SS7)

 **LOCUST**

Host
http://localhost:8000

Status
RUNNING

Users
1


RPS
0.5

Failures
0%

EDIT

STOP

RESET



STATISTICSCHARTSFAILURESEXCEPTIONSCURRENT RATIODOWNLOAD DATALOGS

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	4	0	7	2100	2100	528.74	7	2092	21138	0.5	0
	Aggregated	4	0	7	2100	2100	528.74	7	2092	21138	0.5	0

2026-01-20T09:36:06Z
[2026-01-20 15:06:06,862] lenovo_diya/INFO/locust.main: Shutting down (exit code 0)

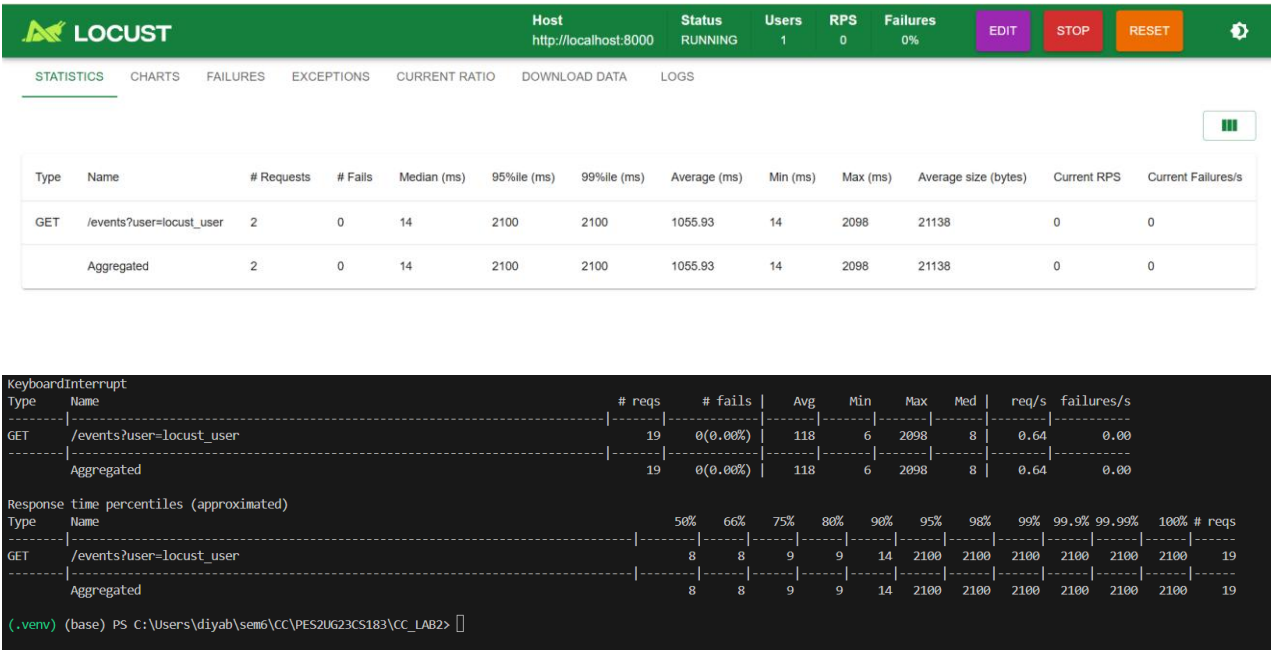
Type	Name	# reqs	# fails	Avg	Min	Max	Med	req/s	failures/s
GET	/events?user=locust_user	16	0(0.00%)	138	6	2092	8	0.70	0.00
	Aggregated	16	0(0.00%)	138	6	2092	8	0.70	0.00

Response time percentiles (approximated)

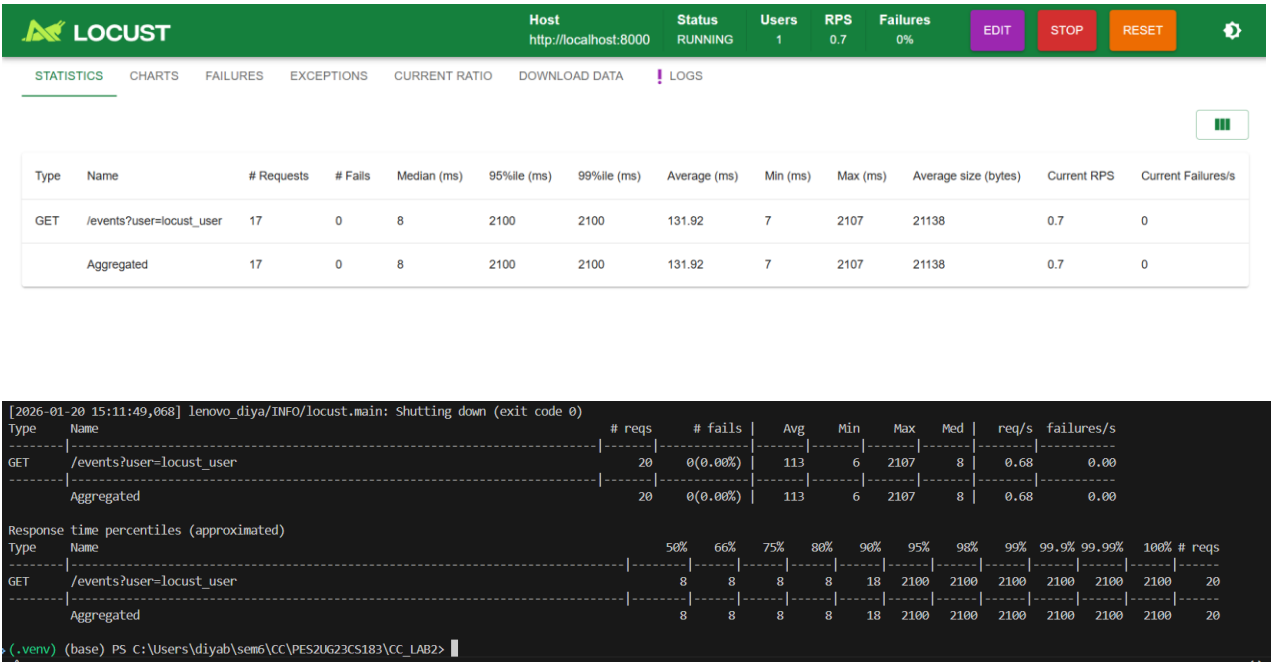
Type	Name	50%	66%	75%	80%	90%	95%	98%	99%	99.9%	99.99%	100%	# reqs
GET	/events?user=locust_user	8	8	9	9	10	2100	2100	2100	2100	2100	2100	16
	Aggregated	8	8	9	9	10	2100	2100	2100	2100	2100	2100	16

(.venv) (base) PS C:\Users\diyab\sem6\CC\PES2UG23CS183\CC_LAB2>

Screenshot 8 (SS8)



Screenshot 9 (SS9)



3. Short explanation of optimizations

Route: /events

What was the bottleneck?

The /events route had an unnecessary loop that ran lot of times and did not contribute to displaying the events. This caused extra CPU usage and slowed down the response.

What change did you make?

I commented the redundant loop from the /events route and then executed it.

Why did the performance improve?

By commenting the unnecessary computation, the server handled requests faster, reducing the response time under load.

Route: /my-events**What was the bottleneck?**

The /my-events route contained a dummy loop with a large number of iterations, which added delay without affecting the output.

What change did you make?

I commented the unused loop from the /my-events route logic and then executed it.

Why did the performance improve?

Commenting the extra loop reduced CPU overhead, resulting in faster request processing and improved response time.

Github Link

https://github.com/PES2UG23CS183/Cloud_Computing_Lab_Diya/tree/main/Week%20%20-%20Monolithic%20Architecture