

# LAB 2 - MONOLITHIC ARCHITECTURE

SRN: PES1UG24CS801  
NAME: ABHISHEK M HIREMATH

SS1

The screenshot shows a web application interface for 'Fest Monolith' with the URL `localhost:8000/events?user=PES1UG24CS801`. At the top, there is a navigation bar with links for 'Events', 'My Events', 'Checkout', and 'Logout'. The main content area is titled 'Events' and displays a grid of twelve event cards. Each card includes an event ID, price, event name, a brief description, and a 'Register' button.

Event ID	Price	Event Name	Description	Action
1	₹ 500	Hackathon	Includes certificate • instant registration • limited seats	Register
2	₹ 300	Dance	Includes certificate • instant registration • limited seats	Register
3	₹ 500	Hackathon	Includes certificate • instant registration • limited seats	Register
4	₹ 300	Dance Battle	Includes certificate • instant registration • limited seats	Register
5	₹ 400	AI Workshop	Includes certificate • instant registration • limited seats	Register
6	₹ 200	Photography Walk	Includes certificate • instant registration • limited seats	Register
7	₹ 350	Gaming Tournament	Includes certificate • instant registration • limited seats	Register
8	₹ 250	Music Night	Includes certificate • instant registration • limited seats	Register
9	₹ 150	Treasure Hunt	Includes certificate • instant registration • limited seats	Register
10	₹ 300	Stand-up Comedy	Includes certificate • instant registration • limited seats	Register
11	₹ 450	Robo Race	Includes certificate • instant registration • limited seats	Register
12	₹ 500	Hackathon	Includes certificate • instant registration • limited seats	Register

SS2

The screenshot shows a web application interface for 'Fest Monolith' with the URL `localhost:8000/checkout`. At the top, there is a navigation bar with links for 'Events', 'My Events', 'Checkout', and 'Logout'. The main content area features a large red box with the heading 'Monolith Failure' and the subtext 'One bug in one module impacted the entire application.' It contains sections for 'Error Message' (showing 'division by zero'), 'Why did this happen?' (explaining it's a monolithic application), and 'What should you do in the lab?' (listing three steps). At the bottom, there are 'Back to Events' and 'Login' buttons.

CC Week X • Monolithic Applications Lab

```
INFO:      127.0.0.1:43716 - "GET /events?user=PES1UG24CS801 HTTP/1.1" 200 OK
INFO:      127.0.0.1:50172 - "GET /checkout HTTP/1.1" 500 Internal Server Error
ERROR:     Exception in ASGI application
Traceback (most recent call last):
```

SS3

8000/checkout

 Fest Monolith  
FastAPI + SQLite + Locust

Logged in as PES1UG24CS801 [Events](#) [My Events](#) [Checkout](#) [Logout](#)

## 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).

```
INFO: 127.0.0.1:52218 - "GET /my-events?user=PES1UG24CS801 HTTP/1.1" 200 OK  
INFO: 127.0.0.1:52218 - "GET /checkout HTTP/1.1" 200 OK
```

SS4

main.py

```
CC Lab-2 > checkout > __init__.py > checkout_logic
1 from database import get_db
2
3 def checkout_logic():
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL ... JavaSE-25 LTS - CC Lab-2 + v ...
```

KeyboardInterrupt  
2026-01-29T09:13:50Z  
[2026-01-29 14:43:50,073] PES1UG24CS801/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%)	3	2	5	3	0.67	0.00	
		Aggregated	20	0 (0.00%)	3	2	5	3	0.67	0.00

Response time percentiles (approximated)

Type	Name	50%	66%	75%	80%	90%	95%	98%	99%	99.9%	
GET	/checkout	3	3	4	4	4	6	6	6	6	20

localhost:8089

Host http://localhost:8000 Status STOPPED RPS 0.6 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)	Cur RPS
GET	/checkout	20	0	3	6	6	3.36	2	6	2797	0.6
	Aggregated	20	0	3	6	6	3.36	2	6	2797	0.6

SS5

## Route 1: /events

SS6

SS7

The screenshot shows a terminal window and a browser window side-by-side.

**Terminal:**

```
ahb1@PES1UG24CS801:~/PES1UG24CS801/CC Lab-2$ locust -f main.py --host http://localhost:8000
```

**Browser:**

The browser displays the Locust UI at [localhost:8089](http://localhost:8089). The top navigation bar shows the host as `http://localhost:8000`, status as `RUNNING`, and users as `1`. The Locust logo is prominently displayed.

The main interface includes tabs for **STATISTICS**, **CHARTS**, **FAILURES**, **EXCEPTIONS**, **CURRENT RATIO**, **DOWNLOAD DATA**, and **LOGS**.

**STATISTICS Table:**

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)
GET	/events? user=locust_user	11	0	3.09	15	15	4.73	3	15	21138
Aggregated		11	0	3.09	15	15	4.73	3	15	21138

**CHARTS:** A chart titled "Response time percentiles (approximated)" shows the distribution of response times for the "/events?user=locust\_user" request. The x-axis represents the percentile from 50% to 99%, and the y-axis represents the number of requests. The data points are: 50% (15), 60% (15), 75% (15), 80% (15), 90% (11), 95% (5), 98% (5), 99% (5).

**FAILURES:** No failures are listed.

**EXCEPTIONS:** No exceptions are listed.

**CURRENT RATIO:** The current ratio is 0.7.

**DOWNLOAD DATA:** A link to download the raw data is present.

**LOGS:** A link to view logs is present.

## Bottleneck:

The /events route contained an unnecessary CPU-intensive loop that executed millions of iterations on every request, even though it did not contribute to the response.

## Change made:

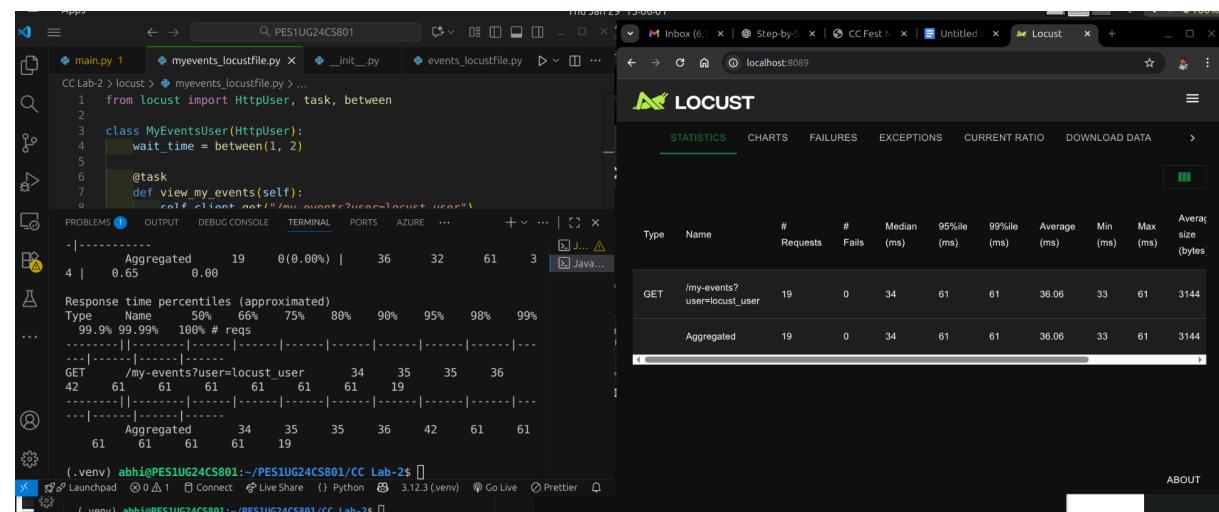
Removed the redundant computation loop and retained only the database query and template rendering logic.

## Why performance improved:

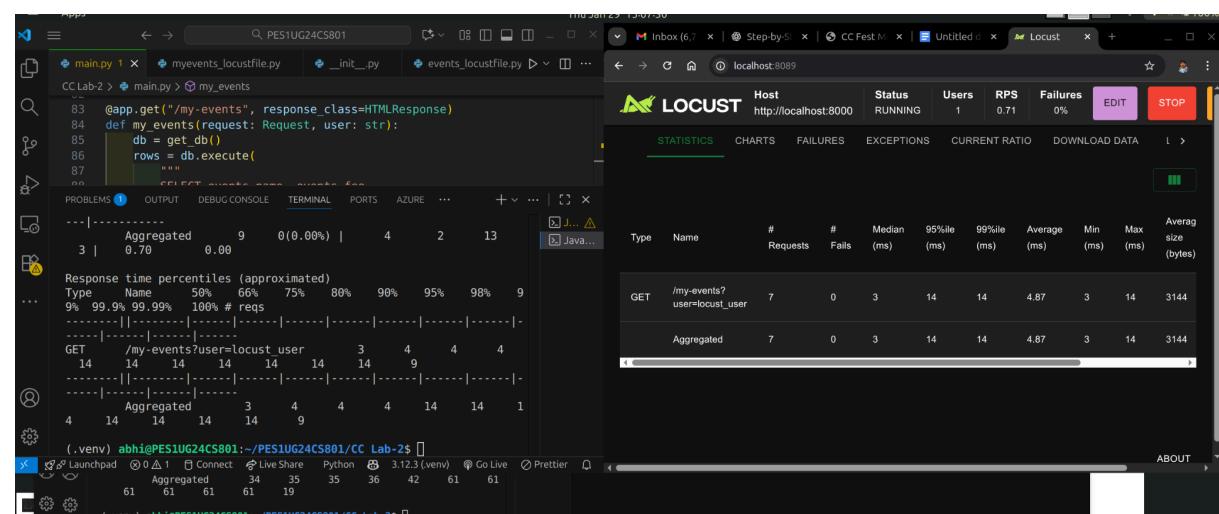
Eliminating unnecessary CPU work reduced processing time per request, leading to lower response time and improved throughput.

## Route 2: /my-events

### SS8



### SS9



**Bottleneck:**

The /my-events route had an artificial delay caused by a large loop that performed meaningless operations for every incoming request.

**Change made:**

Removed the dummy loop and allowed the route to execute only the required database join query and response rendering.

**Why performance improved:**

By removing the extra computation, the server spent less CPU time per request, resulting in faster responses and better overall performance.

**GitHub:-**

<https://github.com/Abhi-DevHub/CC-Lab-2-MONOLITHIC-ARCHITECTURE.git>