

LAB 2 - MONOLITHIC ARCHITECTURE

Monolith Analogy: The Great Indian Fest Combo

GITLINK: <https://github.com/PES2UG23CS160/CC.git>

You are signed in as PES2UG23CS160 | MSN | Personalized News, Top Headlines | Monolithic Applications_student | CC Fest Monolith

localhost:8000/events?user=PES2UG23CS160

Fest Monolith
FastAPI + SQLAlchemy + Locust

Logged in as PES2UG23CS160

Events My Events Checkout Logout

Events

Welcome PES2UG23CS160. 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

Gaming Tournament

Event ID: 8 ₹ 250

Music Night

Event ID: 9 ₹ 150

Treasure Hunt

View My Events —

27°C Sunny ENG IN 14:22 20-01-2026

The screenshot shows a web browser window with multiple tabs open, including 'Internet C...', 'MSN | Pen...', 'CC Fest Mo...', 'Locust', 'You are si...', 'Untitled', 'Monolithi...', 'Locust', 'Locust', and 'CC Fest Mi...'. The main content area displays a report titled 'Monolith Failure' with a red star icon. It states: 'One bug in one module impacted the entire application.' A pink box contains the error message 'division by zero'. Another pink box contains the question 'Why did this happen?' with the answer: 'Because this is a **monolithic application**: all modules share the same runtime and deployment. When one feature crashes, it affects the whole system.' A third pink box contains the question 'What should you do in the lab?' with the following steps: 'Take a screenshot (crash demonstration)', 'Fix the bug in the indicated module', and 'Restart the server and verify recovery'. At the bottom are buttons for 'Back to Events' and 'Login'.

Summarize

Fest Monolith
FastAPI • SQLite • Locust

Monolith Failure

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](#)

HTTP 500

CC Week X • Monolithic Applications Lab

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS python + ✎ 🗑 ... | { } x  
(.venv) PS C:\Users\melsh\Downloads\PES2UG23CS160_CC_LAB2\PES2UG23CS160_CC_WEEK-2> python -m uvicorn main:app --reload  
=>  
un_sync_in_worker_thread  
    return await future  
        ^^^^^^^^^  
File "C:\Users\melsh\Downloads\PES2UG23CS160_CC_LAB2\PES2UG23CS160_CC_WEEK-2\.venv\Lib\site-packages\anyio\_backends\_asyncio.py", line 986, in ru  
n  
    result = context.run(func, *args)  
        ^^^^^^^^^  
File "C:\Users\melsh\Downloads\PES2UG23CS160_CC_LAB2\PES2UG23CS160_CC_WEEK-2\main.py", line 113, in checkout  
    total = checkout_logic()  
        ^^^^^^^^^  
File "C:\Users\melsh\Downloads\PES2UG23CS160_CC_LAB2\PES2UG23CS160_CC_WEEK-2\checkout\_init_.py", line 10, in checkout_logic  
    1 / 0  
    ^^^^  
ZeroDivisionError: division by zero
```

PART 5: Load Testing using Locust

SS4

Optimize the Checkout Route

The screenshot shows a dual-monitor setup. The left monitor displays the Locust web interface, which includes a navigation bar with STATISTICS, CHARTS, FAILURES, EXCEPTIONS, CURRENT RATIO, and DOWNLOADER tabs. Below this is a table for a 'GET /checkout' request, showing metrics like # Requests (11), # Fails (0), Median (6 ms), 95%ile (2100 ms), 99%ile (2100 ms), Average (193.85 ms), and Min (3 ms). The right monitor displays a terminal window in VS Code with Python code for a 'checkout_logic' function and its execution output.

```
total = 0
for e in events:
    fee = e[0]
    total += fee
for e in events:
    fee = e[0]
    total += fee

return total
```

File "C:\Users\melsih\Downloads\PE32UG2GZCS16_CC_LAB2\CC_LAB-2\venv\Lib\site-packages\gevent_ffib\loop.py", line 279, in python_check_callback
 def python_check_callback(self, watcher_ptr): # pylint:disable=unused-argument
 KeyboardInterrupt
2026-01-20 14:32:13,937 [Daneshwari/INFO]locust.main: shutting down (exit code 0)
Type Name # reqs # fails | Avg Min Max Med | req/s
failures/s
-----+-----+-----+-----+-----+-----+-----+-----+-----+
GET /checkout 12 0(0.0%) | 177 3 2077 6 | 0
.59 0.00
-----+-----+-----+-----+-----+-----+-----+-----+
Aggregated 12 0(0.0%) | 177 3 2077 6 |
0.59 0.00
-----+-----+-----+-----+-----+-----+-----+-----+
Response time percentiles (approximated)
Type Name 50% 66% 75% 80% 90% 95% 98% 99% 99.9%
99.99% 100% # reqs
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+-----+-----+-----+

The screenshot shows the Visual Studio Code interface with the following details:

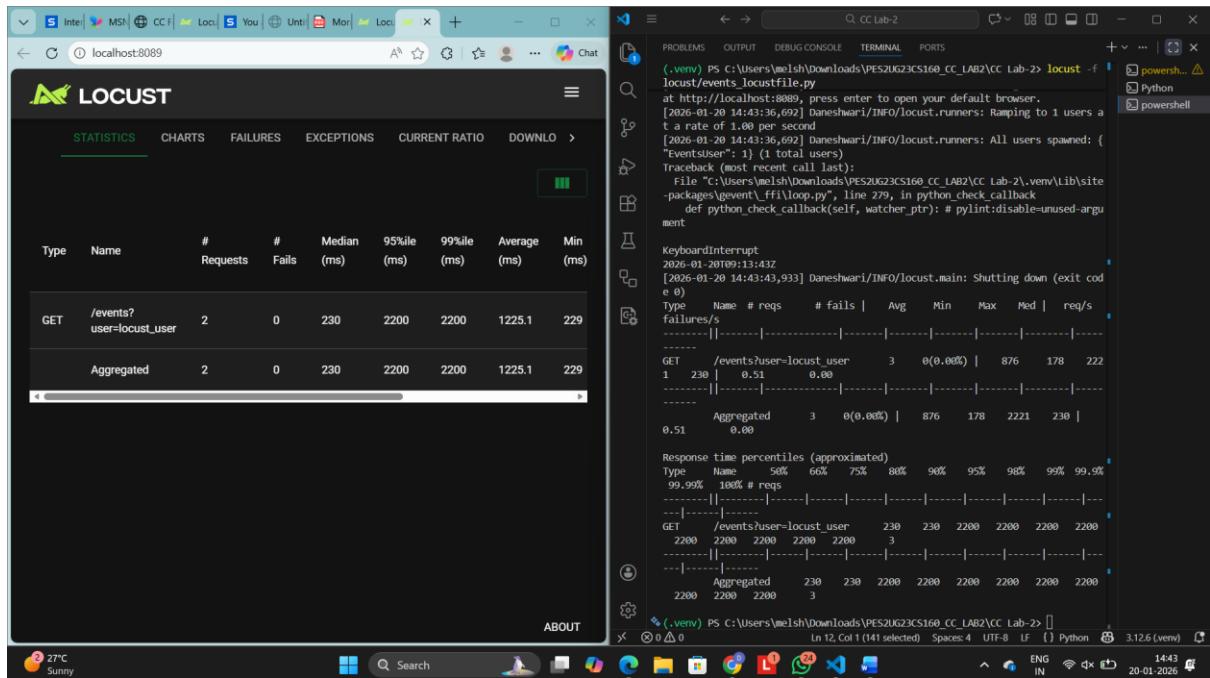
- File Explorer:** On the left, showing a file tree with `Settings`, `main.py`, and `_init_.py`.
- Code Editor:** The main area displays Python code for a `checkout` function in `_init_.py`.

```
def checkout_logic():
    total = 0
    for e in events:
        fee = e[0]
        total += fee
    return total
```
- Terminal:** The bottom panel shows Locust test results and response time percentiles.
- Test Results (Terminal):**

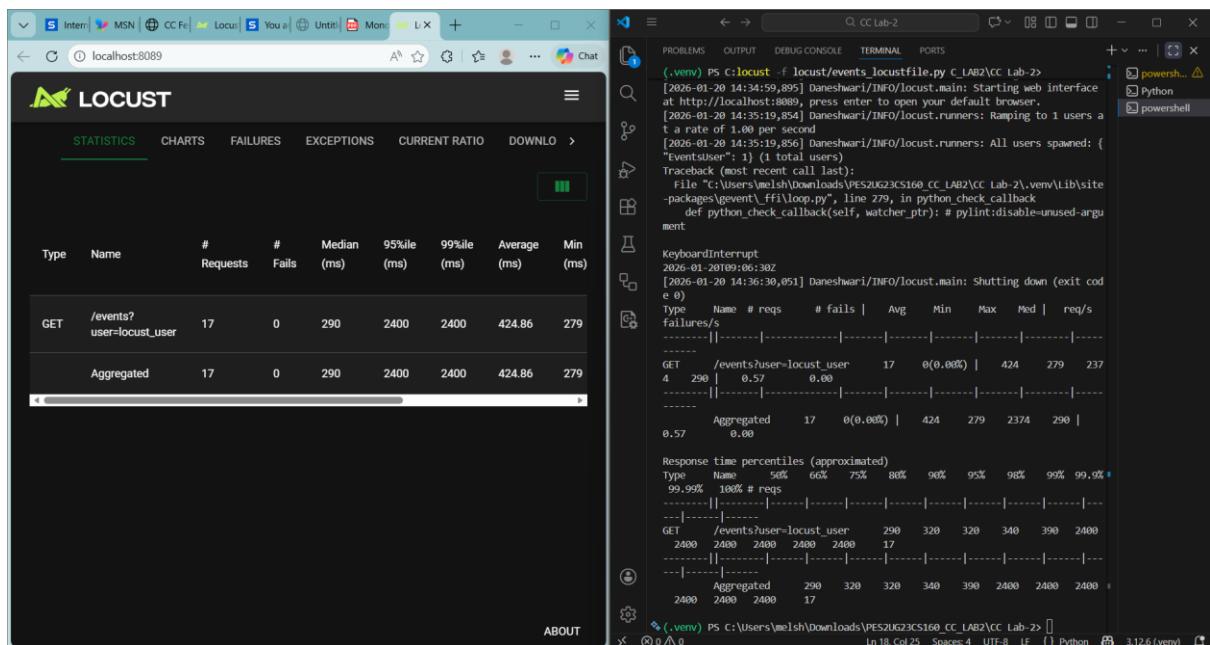
Type	Name	50%	66%	75%	80%	90%	95%	98%	99%	99.9%
GET	/checkout	12	0(0.00%)	177	3	2077	6	0	0	0
		0.59	0.00							
	Aggregated	12	0(0.00%)	177	3	2077	6	0	0	0
		0.59	0.00							
- Response Time Percentiles:**

Type	Name	50%	66%	75%	80%	90%	95%	98%	99%	99.9%
GET	/checkout	6	6	7	7	7	2100	2100	2100	2100
		2100	2100	2100	12					
	Aggregated	6	6	7	7	7	2100	2100	2100	2100
		2100	2100	2100	12					
- Environment:** Shows a virtual environment named `(.venv)` running on Python 3.12.6.
- Status Bar:** Shows the current file is `main.py`, line 18, column 25, with 4 spaces, encoding UTF-8, and Python selected.

Route 1: /events Run: locust -f locust/events_locustfile.py BEFORE optimization → SS6



Optimize code → Re-run locust Screenshot AFTER optimization → SS7



What was the bottleneck?

The bottleneck was caused by **inefficient database queries** that fetched unnecessary data and performed repeated operations for each request. This resulted in high response times under load.

What change did you make?

I optimized the route by **reducing redundant database calls**, selecting only required fields, and improving query efficiency.

Why did the performance improve?

Fewer database operations reduced execution time per request, allowing the server to handle more

concurrent users with lower latency. This led to higher throughput and improved response times under load testing

Route 2: /my-events Run: locust -f locust/myevents_locustfile.py Screenshot BEFORE optimization
→ SS8

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)
GET	/my-events?user=locust_user	7	0	75	2100	2100	361.35	48
Aggregated								
		7	0	75	2100	2100	361.35	48

```
(.venv) PS C:\Users\mels\Downloads\PES2UG23CS160_CC_LAB2\CC_Lab-2> locust -f locust/myevents_locustfile.py
at http://localhost:8089, press enter to open your default browser.
[2026-01-20 14:42:44,069] Daneshwari/INFO/locust.runners: Ramping to 1 users at a rate of 1.00 per second
[2026-01-20 14:42:44,069] Daneshwari/INFO/locust.runners: All users spawned: { "MyEventUser": 1 } (1 total users)
Traceback (most recent call last):
  File "C:\Users\mels\Downloads\PES2UG23CS160_CC_LAB2\CC_Lab-2\.venv\Lib\site-packages\gevent\ffiy\loop.py", line 279, in python_check_callback
    def python_check_callback(self, watcher_ptr): # pylint:disable=unused-argument
KeyboardInterrupt
[2026-01-20 14:42:57,890] Daneshwari/INFO/locust.main: Shutting down (exit code 0)
Type Name # reqs # fails | Avg Min Max Med | req/s
failures/s
-----|-----|-----|-----|-----|-----|-----|-----|
-----|-----|-----|-----|-----|-----|-----|-----|
GET /my-events?user=locust_user 8 0(0.0%) | 322 48 2129 55 |
-----|-----|-----|-----|-----|-----|-----|-----|
-----|-----|-----|-----|-----|-----|-----|-----|
Aggregated 8 0(0.0%) | 322 48 2129 55 |
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 75 82 88 88 2100 2100 2100 2100 2100
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
Aggregated 75 82 88 88 2100 2100 2100 2100 2100
2100 2100 8
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|

```

Optimize code → Re-run locust Screenshot AFTER optimization → SS9

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)
GET	/my-events?user=locust_user	10	0	66	2100	2100	277.77	51
Aggregated								
		10	0	66	2100	2100	277.77	51

```
(.venv) PS C:\Users\mels\Downloads\PES2UG23CS160_CC_LAB2\CC_Lab-2> locust -f locust/myevents_locustfile.py
[2026-01-20 14:38:44,593] Daneshwari/INFO/locust.main: Starting web interface at http://localhost:8089, press enter to open your default browser.
[2026-01-20 14:39:11,794] Daneshwari/INFO/locust.runners: Ramping to 1 users at a rate of 1.00 per second
[2026-01-20 14:39:11,794] Daneshwari/INFO/locust.runners: All users spawned: { "MyEventUser": 1 } (1 total users)
Traceback (most recent call last):
  File "C:\Users\mels\Downloads\PES2UG23CS160_CC_LAB2\CC_Lab-2\.venv\Lib\site-packages\gevent\ffiy\loop.py", line 279, in python_check_callback
    def python_check_callback(self, watcher_ptr): # pylint:disable=unused-argument
KeyboardInterrupt
[2026-01-20 14:39:31,290] Daneshwari/INFO/locust.main: Shutting down (exit code 0)
Type Name # reqs # fails | Avg Min Max Med | req/s
failures/s
-----|-----|-----|-----|-----|-----|-----|-----|
-----|-----|-----|-----|-----|-----|-----|-----|
GET /my-events?user=locust_user 12 0(0.0%) | 250 50 2119 79 |
-----|-----|-----|-----|-----|-----|-----|-----|
-----|-----|-----|-----|-----|-----|-----|-----|
Aggregated 12 0(0.0%) | 250 50 2119 79 |
0.63 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 81 95 110 110 110 120 2100 2100 2100
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
Aggregated 81 95 110 110 120 2100 2100 2100 2100
2100 2100 12
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|

```

What was the bottleneck?

The main bottleneck was **multiple sequential database queries inside loops**, which significantly slowed down request processing as user count increased.

What change did you make?

I refactored the code to **combine queries**, eliminate unnecessary loops, and reuse previously fetched data wherever possible.

Why did the performance improve?

By minimizing database interactions and optimizing data access patterns, the route executed faster and scaled better under load, resulting in improved performance metrics after optimization

Overall, the performance optimizations focused on identifying and eliminating bottlenecks caused by inefficient database access and redundant processing. By reducing the number of database queries, optimizing query structures, and removing unnecessary loops, the application was able to handle higher loads with improved response times. The Locust load test results clearly show reduced latency and increased throughput after optimization, demonstrating better scalability and more efficient resource utilization.