

CC LAB – 2

Name	Dhanya Prabhu
SRN	PES2UG23CS169
Section	C
Date	13/01/2026

PART 1: Setup & Run

```
PS C:\Dhanya\PESENGINEERING\sem6\CC\CC Lab> mkdir PES2UG23CS169

Directory: C:\Dhanya\PESENGINEERING\sem6\CC\CC Lab

Mode                LastWriteTime         Length Name
----                -----          ----  --
d----- 20-01-2026 02:11 PM           0   PES2UG23CS169

PS C:\Dhanya\PESENGINEERING\sem6\CC\CC Lab> python -m venv .venv
PS C:\Dhanya\PESENGINEERING\sem6\CC\CC Lab> .\.venv\Scripts\activate
(.venv) PS C:\Dhanya\PESENGINEERING\sem6\CC\CC Lab\PES2UG23CS169\CC Lab-2> python .\insert_events.py
[✓] Events inserted successfully!
```

PART 2: Use the Application

Screenshot 1 (SS1):

The screenshot shows the Fest Monolith application interface. At the top, there is a navigation bar with the logo 'Fest Monolith' (containing 'cc'), 'FastAPI • SQLite • Locust', and links for 'Events', 'My Events', 'Checkout', and 'Logout'. The user is logged in as 'PES2UG23CS169'. Below the navigation bar, there is a section titled 'Events' with a sub-section header 'Hackathon'. It lists three events:

- Event ID: 1, ₹ 500, Hackathon, Includes certificate • instant registration • limited seats, Register button.
- Event ID: 2, ₹ 300, Dance, Includes certificate • instant registration • limited seats, Register button.
- Event ID: 3, ₹ 500, Hackathon, Includes certificate • instant registration • limited seats, Register button.

Below the Hackathon section, there are two more sections:

- Event ID: 4, ₹ 300, Dance Battle, Includes certificate • instant registration • limited seats, Register button.
- Event ID: 5, ₹ 400, AI Workshop, Includes certificate • instant registration • limited seats, Register button.
- Event ID: 6, ₹ 200, Photography Walk, Includes certificate • instant registration • limited seats, Register button.

On the right side of the screen, there is a link 'View My Events →'.

PART 3: Observe Monolithic Failure (Crash)

Screenshot 2 (SS2):

The screenshot shows a dark-themed web application interface. At the top, there's a header with the text "Fest Monolith" and "FastAPI + SQLite + Locust". On the right side of the header are "Login" and "Create Account" buttons. Below the header, a large red banner displays the text "Monolith Failure" with a star icon. It states "One bug in one module impacted the entire application." A red box highlights an "Error Message" which reads "division by zero". To the right, a red box indicates an "HTTP 500" error. Below these sections, there are two boxes: "Why did this happen?" and "What should you do in the lab?". The "Why did this happen?" box explains that because it's a monolithic application, all modules share the same runtime and deployment, so when one feature crashes, it affects the whole system. The "What should you do in the lab?" box lists three steps: Take a screenshot (crash demonstration), Fix the bug in the indicated module, and Restart the server and verify recovery. At the bottom left are "Back to Events" and "Login" buttons. The footer contains the text "CC Week X • Monolithic Applications Lab" and a log output:

```
INFO: 127.0.0.1:64371 - "GET /events?user=PES2UG23CS169 HTTP/1.1" 200 OK
INFO: 127.0.0.1:64371 - "GET /checkout HTTP/1.1" 500 Internal Server Error
ERROR: Exception in ASGI application
```

PART 4: Fix the Bug

Screenshot 3 (SS3):

The screenshot shows a dark-themed web application interface. At the top, there's a header with the text "Fest Monolith" and "FastAPI + SQLite + Locust". On the right side of the header are "Login" and "Create Account" buttons. Below the header, there's a section titled "Checkout" with a blue bar icon. It says "This route is used to demonstrate a monolith crash + optimization." A box shows "Total Payable ₹ 6600". Below this, a green box contains the text "After fixing + optimizing checkout logic, re-run Locust and compare results." To the right, a section titled "What you should observe" lists three points: One buggy feature can crash the entire monolith, Inefficient loops cause high response times under load, and Optimization improves performance but architecture still scales as one unit. A yellow box at the bottom right says "Next Lab: Split this monolith into Microservices (Events / Registration / Checkout)." The footer contains the text "CC Week X • Monolithic Applications Lab" and a log output:

```
INFO: 127.0.0.1:49805 - "GET /events?user=PES2UG23CS169 HTTP/1.1" 200 OK
INFO: 127.0.0.1:49805 - "GET /checkout HTTP/1.1" 200 OK
```

PART 5: Load Testing using Locust

Screenshot 4 (SS4):

LOCUST

Host: http://localhost:8000 Status: CLEANUP RPS: 0.8 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	18	0	11	2100	2100	132.57	5	2132	2797	0.8	0
Aggregated												
GET	/checkout	18	0	11	2100	2100	132.57	5	2132	2797	0.8	0

```
PS C:\Dhanya\PESENGINEERING\sem6\CC\CC Lab\PES2UG23CS169\CC Lab-2> python -m locust -f locust\checkout_locustfile.py
[2026-01-20 14:33:44,384] DhanyaPrabhu/INFO/locust.main: Starting Locust 2.43.1
[2026-01-20 14:33:44,386] DhanyaPrabhu/INFO/locust.main: Starting web interface at http://localhost:8089, press enter to open your default browser.
[2026-01-20 14:35:45,273] DhanyaPrabhu/INFO/locust.runners: Ramping to 1 users at a rate of 1.00 per second
[2026-01-20 14:35:45,275] DhanyaPrabhu/INFO/locust.runners: All users spawned: {"CheckoutUser": 1} (1 total users)
Traceback (most recent call last):
  File "C:\Users\dhany\AppData\Roaming\Python\Python313\site-packages\gevent\_ffi\loop.py", line 279, in python_check_callback
    def python_check_callback(self, watcher_ptr): # pylint:disable=unused-argument

KeyboardInterrupt
2026-01-20T09:06:50Z
[2026-01-20 14:36:50,118] DhanyaPrabhu/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%) | 132     4      2132    11 | 0.62    0.00
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
Aggregated          18      0(0.00%) | 132     4      2132    11 | 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      /checkout      11      12     14     15     82     2100   2100   2100   2100   2100   2100   18
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
Aggregated          11      12     14     15     82     2100   2100   2100   2100   2100   2100   18
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
PS C:\Dhanya\PESENGINEERING\sem6\CC\CC Lab\PES2UG23CS169\CC Lab-2> 
```

PART 6: Optimize the Checkout Route

Screenshot 5 (SS5):

LOCUST

Host: http://localhost:8000 Status: CLEANUP RPS: 0.6 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	10	2100	2100	123.98	6	2100	2797	0.6	0
Aggregated												
GET	/checkout	19	0	10	2100	2100	123.98	6	2100	2797	0.6	0

```
PS C:\Dhanya\PESENGINEERING\sem6\CC\CC Lab\PES2UG23CS169\CC Lab-2> python -m locust -f locust\checkout_locustfile.py
[2026-01-20 14:39:23,190] DhanyaPrabhu/INFO/locust.main: Starting Locust 2.43.1
[2026-01-20 14:39:23,191] DhanyaPrabhu/INFO/locust.main: Starting web interface at http://localhost:8089, press enter to open your default browser.
[2026-01-20 14:39:39,016] DhanyaPrabhu/INFO/locust.runners: Ramping to 1 users at a rate of 1.00 per second
[2026-01-20 14:39:39,018] DhanyaPrabhu/INFO/locust.runners: All users spawned: {"CheckoutUser": 1} (1 total users)
Traceback (most recent call last):
  File "C:\Users\dhany\AppData\Roaming\Python\Python313\site-packages\gevent\_ffi\loop.py", line 279, in python_check_callback
    def python_check_callback(self, watcher_ptr): # pylint:disable=unused-argument

KeyboardInterrupt
2026-01-20T09:10:27Z
[2026-01-20 14:40:27,762] DhanyaPrabhu/INFO/locust.main: Shutting down (exit code 0)
Type      Name          # reqs   # fails | Avg     Min     Max     Med | req/s  failures/s
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
Type      Name          50%    66%    75%    80%    90%    95%    98%    99%    99.9% 99.99% 100% # reqs
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
GET      /checkout      10      11     15     27     43     2100   2100   2100   2100   2100   2100   19
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
Aggregated          10      11     15     27     43     2100   2100   2100   2100   2100   2100   19
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
PS C:\Dhanya\PESENGINEERING\sem6\CC\CC Lab\PES2UG23CS169\CC Lab-2> 
```

PART 7: Optimise events and my_events(DIY)

Screenshot 6 (SS6):

Locust

Host: http://localhost:8000

Status: CLEANUP RPS: 0.4 Failures: 0%

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	14	0	460	2600	2600	604.12	358	2631	21138	0.4	0
Aggregated		14	0	460	2600	2600	604.12	358	2631	21138	0.4	0

```
PS C:\Dhanya\PESENGINEERING\sem6\CC\CC Lab\PES2UG23CS169\CC Lab-2> python -m locust -f locust/events_locustfile.py
[2026-01-20 14:45:23,810] DhanyaPrabhu/INFO/locust.main: Starting Locust 2.43.1
[2026-01-20 14:45:23,812] DhanyaPrabhu/INFO/locust.main: Starting web interface at http://localhost:8089, press enter to open your default browser.
[2026-01-20 14:46:56,841] DhanyaPrabhu/INFO/locust.runners: Ramping to 1 users at a rate of 1.00 per second
[2026-01-20 14:46:56,844] DhanyaPrabhu/INFO/locust.runners: All users spawned: {"EventsUser": 1} (1 total users)
Traceback (most recent call last):
  File "C:\Users\dhany\AppData\Roaming\Python\Python313\site-packages\gevent\_ffi\loop.py", line 279, in python_check_callback
    def python_check_callback(self, watcher_ptr): # pylint:disable=unused-argument

KeyboardInterrupt
2026-01-20T09:17:53Z
[2026-01-20 14:47:53,449] DhanyaPrabhu/INFO/locust.main: Shutting down (exit code 0)
Type      Name          # reqs   # fails | Avg     Min     Max     Med | req/s failures/s
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    470    520    520    530    550    2600   2600   2600   2600   2600   2600   2600   14
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
Aggregated                      470    520    520    530    550    2600   2600   2600   2600   2600   2600   2600   14

PS C:\Dhanya\PESENGINEERING\sem6\CC\CC Lab\PES2UG23CS169\CC Lab-2> 
```

Screenshot 7 (SS7):

Screenshot 8 (SS8):

Screenshot 9 (SS9):

Explanations:

Routes: /events

1. What was the bottleneck?

The /events endpoint had an unnecessary heavy CPU loop running on every request, which increased the response time.

2. What change did you make?

I commented the unnecessary loop so the endpoint only performs the required logic.

3. Why did the performance improve?

Because the server stopped doing extra computations for every request, the response time reduced and the endpoint handled requests faster.

Routes: /my-events

1. What was the bottleneck?

The /my-events endpoint had an unnecessary heavy CPU loop running on every request, which caused delay and increased processing time.

2. What change did you make?

I commented the unnecessary dummy loop to reduce the extra workload during each API call.

3. Why did the performance improve?

Since the endpoint no longer wastes time in redundant operations, it responds quicker and overall performance improves under load.