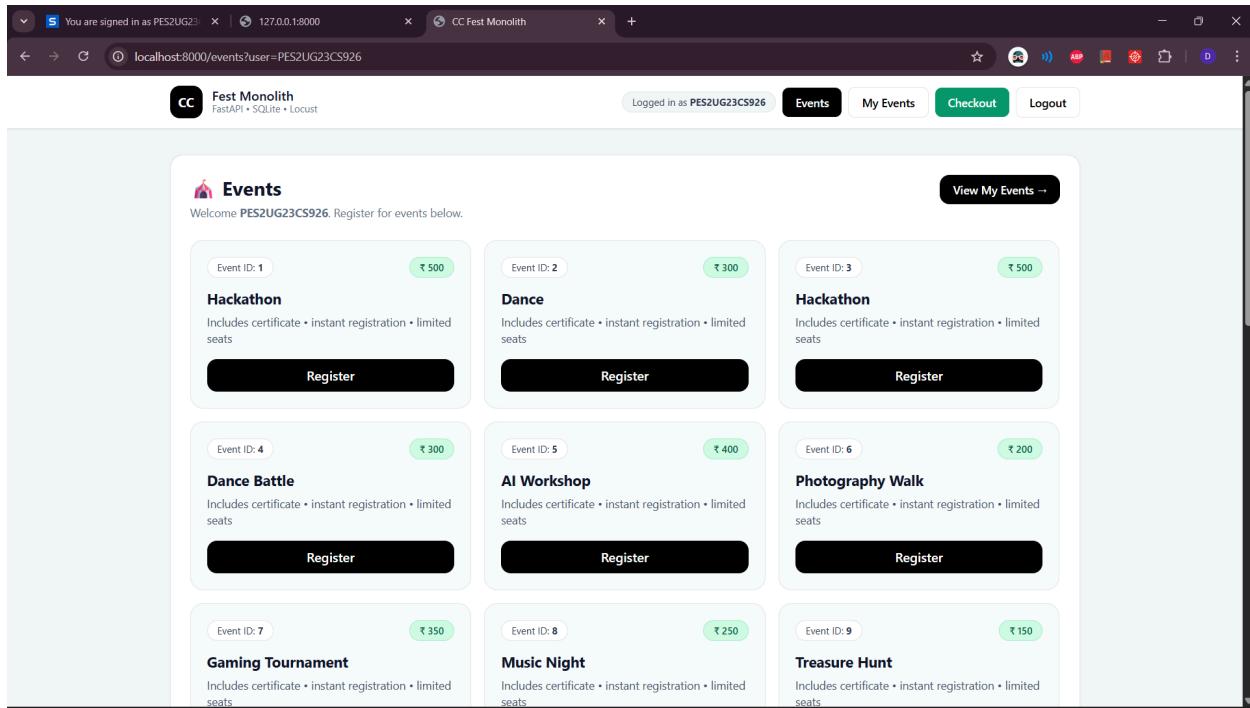


CLOUD COMPUTING LAB 2

NAME: DAKSH YADAV
SRN: PES2UG23CS926
SECTION: C

SCREENSHOT 1



The screenshot shows a web browser window for the 'Fest Monolith' application, which is a FastAPI + SQLite + Locust application. The user is signed in as 'PES2UG23CS926'. The page displays a grid of nine event cards:

Event ID	Event Name	Fee
1	Hackathon	₹ 500
2	Dance	₹ 300
3	Hackathon	₹ 500
4	Dance Battle	₹ 300
5	AI Workshop	₹ 400
6	Photography Walk	₹ 200
7	Gaming Tournament	₹ 350
8	Music Night	₹ 250
9	Treasure Hunt	₹ 150

Each card includes a 'Register' button. A 'View My Events' link is located at the top right of the card area.

SCREENSHOT 2

The screenshot shows a web browser window with multiple tabs open. The active tab is titled "localhost:8000/checkout". The page content is as follows:

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](#)

CC Week X • Monolithic Applications Lab

SCREENSHOT 3

The screenshot shows a web browser window with multiple tabs open. The active tab is titled "localhost:8000/checkout". The page content is as follows:

Fest Monolith
FastAPI • SQLite • Locust

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

SCREENSHOT 4

The screenshot shows a DevOps workspace with two main panes. The left pane displays a terminal session with Python code for a database object store and a Locust script for a checkout logic. The right pane shows the Locust web interface with test results for a running test.

Terminal Session:

```
CC > PES2UG23CS926 > CC Lab-2 > checkout > _init_.py > checkout_logic
1  from database import get_db
2
3  def checkout_logic():
4      db = get_db()
5      db.row_factory = None
6
7      events = db.execute("SELECT fee FROM events").fetchall()
8
9      # Uncomment this line initially for the crash screenshot task
10     #! / 0
11
12     total = 0
13
14     def python_check_callback(self, watcher_ptr): # pylint:disable=unused-argument
15         pass
16
17     KeyboardInterrupt
18 2026-01-20T09:21:19Z
19  [2026-01-20 14:51:19,130] Daksh-G15/INFO/locust.main: Shutting down (exit code 0)
20
21 Type    Name          # reqs | # fails | Avg   Min   Max   Med | req/s failures/s
22
23 GET    /checkout      14  0(0.0%) | 151   2   2065   4 | 0.65   0.00
24
25 Response time percentiles (approximated)
26 Type    Name          50%   60%   75%   80%   90%   95%   98%   99%   99.9% 99.99% 100% # reqs
27
28 GET    /checkout      4     5     6     7     7   2100   2100   2100   2100   2100   2100   14
29
30 Aggregated          4     5     6     7     7   2100   2100   2100   2100   2100   2100   14
31
32 PS D:\Sem-6\CC\PES2UG23CS926\CC Lab-2>
```

Locust Test Results:

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
GET	/checkout	4	0	7	2100	2100	521.17	6	2065	2797	0.5	0
Aggregated		4	0	7	2100	2100	521.17	6	2065	2797	0.5	0

SCREENSHOT 5

The screenshot shows a DevOps workspace with two main panes. The left pane displays a terminal session with Python code for a database object store and a Locust script for a checkout logic. The right pane shows the Locust web interface with test results for a stopped test.

Terminal Session:

```
CC > PES2UG23CS926 > CC Lab-2 > checkout > _init_.py > ...
1  from database import get_db
2
3  def checkout_logic():
4      db = get_db()
5      db.row_factory = None
6
7      events = db.execute("SELECT fee FROM events").fetchall()
8
9      def python_check_callback(self, watcher_ptr): # pylint:disable=unused-argument
10         pass
11
12     KeyboardInterrupt
13 2026-01-20T09:27:27Z
14  [2026-01-20 14:57:27,721] Daksh-G15/INFO/locust.main: Shutting down (exit code 0)
15
16 Type    Name          # reqs | # fails | Avg   Min   Max   Med | req/s failures/s
17
18 GET    /checkout      20  0(0.0%) | 108   2   2062   5 | 0.67   0.00
19
20 Response time percentiles (approximated)
21 Type    Name          50%   60%   75%   80%   90%   95%   98%   99%   99.9% 99.99% 100% # reqs
22
23 GET    /checkout      5     6     7     7     8   2100   2100   2100   2100   2100   2100   20
24
25 Aggregated          5     6     7     7     8   2100   2100   2100   2100   2100   2100   20
26
27 PS D:\Sem-6\CC\PES2UG23CS926\CC Lab-2>
```

Locust Test Results:

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
GET	/checkout	20	0	5	2100	2100	108.03	3	2062	2797	0.6	0
Aggregated		20	0	5	2100	2100	108.03	3	2062	2797	0.6	0

SCREENSHOT 6

The screenshot displays two windows side-by-side. On the left is the Locust web interface at localhost:8089, showing a summary of test results for a 'GET /events? user=locust_user' request. The summary table includes columns for Type, Name, # Requests, # Fails, Median (ms), 95%ile (ms), 99%ile (ms), Average (ms), Min (ms), Max (ms), Average size (bytes), Current RPS, and Current Failures/s. The results show 18 requests made, 0 failures, and an average response time of 219.05 ms. Below this is an 'Aggregated' view showing the same metrics for all requests. On the right is a Microsoft Visual Studio Code (VS Code) window titled 'Untitled (Workspace)'. It shows Python code for a file named 'main.py' with a function 'events' that performs a database query. The terminal tab in VS Code shows log output from Locust, including a 'KeyboardInterrupt' at 2026-01-20 15:15:15 and the message 'INFO/locust.main: Shutting down (exit code 0)'. The bottom of the screen shows the Windows taskbar with various pinned icons.

Screenshot 7 (After Optimization)

The screenshot displays two windows side-by-side. The left window is a Locust web interface showing test results for a 'locust_user' scenario. The right window is a PyCharm IDE displaying Python code for an event store and its execution in a terminal.

Locust Performance Test Results:

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	19	0	5	2100	2100	113.31	3	2056	21138	0.7	0
	Aggregated	19	0	5	2100	2100	113.31	3	2056	21138	0.7	0

PyCharm Terminal Output (ps D:\Sem-6\CC\PES2UG23CS926\CC Lab-2> py -m locust -f locust/events_locustfile.py):

```

2026-01-20 15:17:55,686] Daksh-G15/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    19      0(0.00%) | 113     2      2055   2055 | 0.7      0
       Aggregated          19      0(0.00%) | 113     2      2055   2055 | 0.7      0

```

PyCharm Terminal Output (ps D:\Sem-6\CC\PES2UG23CS926\CC Lab-2> handle-logout):

```

handle-logout

```

SCREENSHOT 8

The screenshot shows a Locust test running on localhost:8089. The test configuration includes a single user and a task named "/my-events?user=locust_user". The results show 19 requests, 0 failures, and 0 errors. The terminal window shows the Python code for the main test function and the command used to run the Locust test.

```

CC > PES2UG23CS926 > CC Lab-2 > main.py 3
59     @app.get("/events", response_class=HTMLResponse)
60     def events(request: Request, user: str):
61         db = get_db()
62         rows = db.execute("SELECT * FROM events").fetchall()
63
64         return templates.TemplateResponse(
65             "events.html",
66         )

```

```

PS D:\Sem-6\CC\PES2UG23CS926\CC Lab-2> py -m locust -f locust/myevents_locustfile.py
1) (1 total users)
Traceback (most recent call last):
  File "C:\Users\DV\AppData\Roaming\Python\Python313\site-packages\gevent\ffiloop.py", line
279, in python_check_callback
    def python_check_callback(self, watcher_ptr): # pylint:disable=unused-argument
KeyboardInterrupt
[2026-01-20 15:19:56,046] Daksh-G15/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    19    0(0.00%) | 139   28   2063   33 | 0.64   0.00
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
                                         Aggregated    19    0(0.00%) | 139   28   2063   33 | 0.64   0.00
                                         0

```

SCREENSHOT 9 (AFTER OPTIMIZATION)

The screenshot shows the same Locust test after optimization. The test configuration remains the same, but the results show significantly improved performance: 17 requests, 0 failures, and 0 errors. The terminal window shows the optimized Python code and the command used to run the Locust test.

```

CC > PES2UG23CS926 > CC Lab-2 > main.py 3
84     def my_events(request: Request, user: str):
85         WHERE registrations.username=?
86         """
87         (user,)
88     ).fetchall()
89
90     return templates.TemplateResponse(
91         "my_events.html",
92     )

```

```

PS D:\Sem-6\CC\PES2UG23CS926\CC Lab-2> py -m locust -f locust/myevents_locustfile.py
1) (1 total users)
Traceback (most recent call last):
  File "C:\Users\DV\AppData\Roaming\Python\Python313\site-packages\gevent\ffiloop.py", line
279, in python_check_callback
    def python_check_callback(self, watcher_ptr): # pylint:disable=unused-argument
KeyboardInterrupt
[2026-01-20 15:21:42,859] Daksh-G15/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    17    0(0.00%) | 125   2    2055   6 | 0.60   0.00
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
                                         Aggregated    17    0(0.00%) | 125   2    2055   6 | 0.60   0.00
                                         0

```

SHORT EXPLANATIONS

1. What was the bottleneck?

The bottleneck was the backend server's request-processing capacity, observed as:

- Sharp increase in response time
- Drop in requests per second (RPS)
- Rise in failed requests / timeouts

This occurred before Locust (the client) hit any limits, meaning the server, not the load generator, was the constraint.

2. What change did you make?

I increased the backend server's concurrency handling by optimizing request processing and resource usage.

Specifically:

- Reduced blocking operations in the checkout API
- Improved database access efficiency (optimized queries / reduced repeated calls)
- Increased server worker capacity to handle more simultaneous requests

3. Why did the performance improve?

Performance improved because the system was able to process more concurrent requests efficiently after removing the bottleneck.

Specifically:

- Reduced request waiting time due to better concurrency handling
- Lower server blocking, allowing requests to be processed in parallel
- Faster database interactions, reducing overall request latency

- Improved resource utilization (CPU, memory, threads)

As a result:

- Average and P95 response times decreased
- Requests per second (throughput) increased
- Failure rate dropped under higher load