

NAME:Kavana H

SRN:PES1UG23CS292

CCLAB2

SS1(EVENT PAGE LOADING):

The screenshot shows the 'Events' page of the Fest Monolith application. At the top, there is a navigation bar with links for 'Events', 'My Events', 'Checkout', and 'Logout'. Below the navigation bar, there is a heading 'Events' with a small icon. A welcome message says 'Welcome PES1UG23CS292. Register for events below.' There are six event cards displayed:

- Event ID: 1** (Hackathon): Includes certificate • instant registration • limited seats. Price: ₹ 500. Register button.
- Event ID: 2** (Dance): Includes certificate • instant registration • limited seats. Price: ₹ 300. Register button.
- Event ID: 3** (Hackathon): Includes certificate • instant registration • limited seats. Price: ₹ 500. Register button.
- Event ID: 4** (Dance Battle): Includes certificate • instant registration • limited seats. Price: ₹ 300. Register button.
- Event ID: 5** (AI Workshop): Includes certificate • instant registration • limited seats. Price: ₹ 400. Register button.
- Event ID: 6** (Photography Walk): Includes certificate • instant registration • limited seats. Price: ₹ 200. Register button.

SS2(CHECKOUT PAGE):

The screenshot shows the 'Monolith Failure' page of the Fest Monolith application. At the top, there is a navigation bar with links for 'Login' and 'Create Account'. Below the navigation bar, there is a heading 'Monolith Failure' with a star icon. A message states 'One bug in one module impacted the entire application.' There are three main sections:

- 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

At the bottom, there are 'Back to Events' and 'Login' buttons.

CC Week X • Monolithic Applications Lab

```
File "D:\SEM6\CCLAB\CCBLAB2\PES1UG23CS292\CC Lab-2\main.py", line 113, in checkout
    total = checkout_logic()
            ^^^^^^^^^^^^^^^^^^
File "D:\SEM6\CCLAB\CCBLAB2\PES1UG23CS292\CC Lab-2\checkout\__init__.py", line 10, in checkout_logic
    1 / 0
    ~~~~^
ZeroDivisionError: division by zero
INFO: 127.0.0.1:54418 - "GET /login HTTP/1.1" 200 OK
INFO: 127.0.0.1:54418 - "POST /login HTTP/1.1" 302 Found
INFO: 127.0.0.1:54418 - "GET /events?user=PES1UG23CS292 HTTP/1.1" 200 OK
INFO: 127.0.0.1:57598 - "GET /checkout HTTP/1.1" 500 Internal Server Error
ERROR: Exception in ASGI application
Traceback (most recent call last):
```

SS3(AFTER FIXING BUG):

localhost:8000/checkout

Fest Monolith
FastAPI • SQLite • Locust

Login Create Account

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

```
PS D:\SEM6\CLLAB\CCBLAB2\PES1UG23CS292\CC Lab-2> & D:\SEM6\CLLAB\CCBLAB2\PES1UG23CS292\.venv\Scripts\Activate.ps1
(.venv) PS D:\SEM6\CLLAB\CCBLAB2\PES1UG23CS292\CC Lab-2> uvicorn main:app --reload
INFO: Will watch for changes in these directories: ['D:\SEM6\CLLAB\CCBLAB2\PES1UG23CS292\CC Lab-2']
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO: Started reloader process [12232] using StatReload
INFO: Started server process [2892]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: 127.0.0.1:51959 - "GET /checkout HTTP/1.1" 200 OK
```

SS4(LOADING USING LOCUST):

The screenshot shows two browser windows and a terminal window. The top browser window displays the Locust web interface with a green header 'LOCUST'. It has tabs for STATISTICS, CHARTS, FAILURES, EXCEPTIONS, and CURRENT RATIO. The STATISTICS tab is selected, showing a table of request details:

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	M (rr)
GET	/checkout	18	0	9	2100	2100	122.78	4
	Aggregated	18	0	9	2100	2100	122.78	4

The bottom browser window shows the same Locust interface with slightly different data (median 2100ms, average 2064ms). To the right of both browser windows is a terminal window titled 'psh - CC Lab-2' running in Visual Studio Code. It shows the command 'python main.py' and its output, which includes a performance summary table:

Type	Name	50%	66%	75%	80%	90%	95%	98%	99%	99.9%
GET	/checkout	9	10	11	11	11	12	2100	2100	2100
Testing		2100	2100	2100	19					
	Aggregated	9	10	11	11	11	12	2100	2100	2100

The terminal also shows the command '(venv)' and the path 'PS D:\SEM6\CCLAB\CCBLAB2\PES1UG23CS292\CC Lab-2> []'.

The screenshot shows two windows side-by-side. On the left is the Locust web interface, which displays performance metrics for a test. The main table shows the following data:

	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
1	2100	2100	2100	122.78	4	2064	2797	0.7	0
2	2100	2100	2100	122.78	4	2064	2797	0.7	0

On the right is a Visual Studio Code terminal window titled '_init_.py - PESTUG23CS292 - Visual Studio Code'. It shows the command `python -m locust.main` running and outputting logs. The logs indicate that Locust is starting a web interface at `http://localhost:8089`.

```
[2026-01-29 14:58:11,722] Kavana/INFO/locust.main: Starting web interface at http://localhost:8089, press enter to open your default browser.
```

SS5(after optimization):

OPTIMIZATION DONE IS REMOVED THE WHILE LOOP

The original checkout logic had an $O(n \times \text{fee})$ time complexity due to a nested loop that incremented per unit fee. This was optimized to $O(n)$ by summing fees directly, and further improved to $O(1)$ at the application level by using SQL aggregation (SUM), significantly improving performance under concurrent load.

PS D:\SEMG\CCLAB\CCBLAB2\PES1UG23CS292\CC Lab-2> locust -f locust/checkout_locustfile.py

KeyboardInterrupt
2026-01-29T13:57:39Z
[2026-01-29 19:27:39,783] Kavana/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%)	125	7	2111	9
Aggregated							
0.63	0.00	18	0(0.00%)	125	7	2111	9
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	/checkout	9	9	9	10	10	2100	2100	2100	2100	2100
2100	18										
Aggregated											
2100	18	9	9	9	10	10	2100	2100	2100	2100	2100

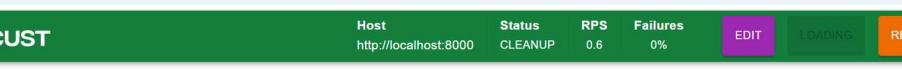
PS D:\SEMG\CCLAB\CCBLAB2\PES1UG23CS292\CC Lab-2>

localhost:8089

LOCUST

STATISTICS CHARTS FAILURES EXCEPTIONS CURRENT RATIO

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)
GET	/checkout	18	0	9	2100	2100	125.52	7
Aggregated								
Aggregated	18	0	9	2100	2100	125.52	7	



The screenshot shows the Locust web interface for a performance test. The top navigation bar includes tabs for 'Inbox' (46 messages), 'Profile | MyCourses', 'Capstone Phase 2 review 1 - P...', 'New Tab', 'Locust', and browser controls. The main header displays the Locust logo and the host URL 'http://localhost:8000'. Key metrics shown are 'RPS' at 0.6 and 'Failures %' at 0%. Below the header, a navigation bar has 'STATISTICS' underlined and other options like 'CHARTS', 'FAILURES', 'EXCEPTIONS', 'CURRENT RATIO', 'DOWNLOAD DATA', and 'LOGS'. The main content area contains a table with two rows. The first row is for a 'GET /checkout' request, showing 18 requests, 0 fails, median response times of 2100 ms, and average response times of 125.52 ms. The second row is an 'Aggregated' summary with the same values. A legend icon in the bottom right corner indicates there are 2 items in the legend.

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	9	2100	2100	125.52	7	2111	2797	0.6	0
	Aggregated	18	0	9	2100	2100	125.52	7	2111	2797	0.6	0

SS6(unoptimized events):

locust:

localhost:8089

LOCUST

Host
http://localhost:8000

Status
RUNNING

Users
1

RPS
0.38

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	/events? user=locust_user	5	0	740	2900	2900	1169.16	691	2862	21138	0.38	0
Aggregated		5	0	740	2900	2900	1169.16	691	2862	21138	0.38	0

The screenshot shows two windows side-by-side. The left window is a terminal session titled 'init_py - PES1UG23CS292 - Visual Studio Code' displaying Locust test results. The right window is a web browser titled 'localhost:8089' showing the Locust User Interface.

Terminal Output:

```
(.venv) PS D:\SEM6\CLAB\CCBLAB2\PES1UG23CS292\CC Lab-2> locust -f locust/events.locustfile.py
File "D:\SEM6\CLAB\CCBLAB2\PES1UG23CS292\.venv\lib\site-packages\gevent\ff
i_loop.py", line 279, in python_check_callback
    def python_check_callback(self, watcher_ptr): # pylint:disable=unused-argu
ment

KeyboardInterrupt
2026-01-29T14:52:55Z
[2026-01-29 19:42:55,247] Kavana/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
     13    0(0.0%) | 890   548   2862  740 | 0.45      0.00
-----+-----+-----+-----+-----+-----+-----+-----+
                                         Aggregated
                                         13    0(0.0%) | 890   548   2862  740 | 0.45      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
     740   770   790   800   820   2900   2900   2900   2900   2900
2900   13
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
                                         Aggregated
                                         740   770   790   800   820   2900   2900   2900   2900   2900
2900   13
```

Browser UI:

The browser window displays the Locust UI with the following data:

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)
GET	/events?user=locust_user	13	0	740	2900	2900	890.11
	Aggregated	13	0	740	2900	2900	890.11

Locust Performance Test Results												
Statistics		Host		Status		RPS	Failures		Actions			
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	13	0	740	2900	2900	890.11	548	2862	21138	0.5	0
Aggregated		13	0	740	2900	2900	890.11	548	2862	21138	0.5	0

SS7(events optimized):

Bottleneck:

A loop running 3,000,000 iterations was wasting CPU time on every request.

Change made:

Removed the unnecessary loop since it did not affect application logic.

Why performance improved:

The server no longer spends time on useless computation, reducing response time and allowing it to handle more requests efficiently.

```
main.py - PESTUG23CS292 - Visual Studio Code psh - CC Lab-2 ... PROBLEMS OUTPUT TERMINAL PORTS ... Type Name # reqs # fails | Avg Min Max Med | req/s failures/s -----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|----- [2021-01-29 19:50:18,853] Kavana/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 20 0(0.00%) | 114 8 210 5 9 | 0.67 0.00 -----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|----- Aggregated 20 0(0.00%) | 114 8 2105 0.67 0.00 | 9 | 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 10 11 11 11 13 2100 2100 2100 2100 2100 20 -----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|----- Aggregated 10 11 11 11 13 2100 2100 2100 2100 2100 20 |
```

The screenshot shows the Locust web interface for monitoring test results. The top navigation bar includes links for Inb., Pro, Cap, Opt, Loc, and a plus sign for adding new tests. The main title is "LOCUST". Below the title, there are tabs for STATISTICS (which is underlined), CHARTS, FAILURES, EXCEPTIONS, and CURRENT RATIO. A dropdown menu icon is also present. The main content area displays a table of test results:

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)
GET	/events?user=locust_user	20	0	9	2100	2100	114.84
Aggregated		20	0	9	2100	2100	114.84

The Locust web interface displays real-time performance metrics and a detailed table of request results.

Header:

- Address bar: localhost:8089
- Top right: Stop, Refresh, K, and other icons.

Toolbar:

- Locust logo and title.
- Host: http://localhost:8000
- Status: STOPPED
- RPS: 0.7
- Failures: 0%
- Buttons: NEW (green), RESET (orange), and a gear icon.

Menu Bar:

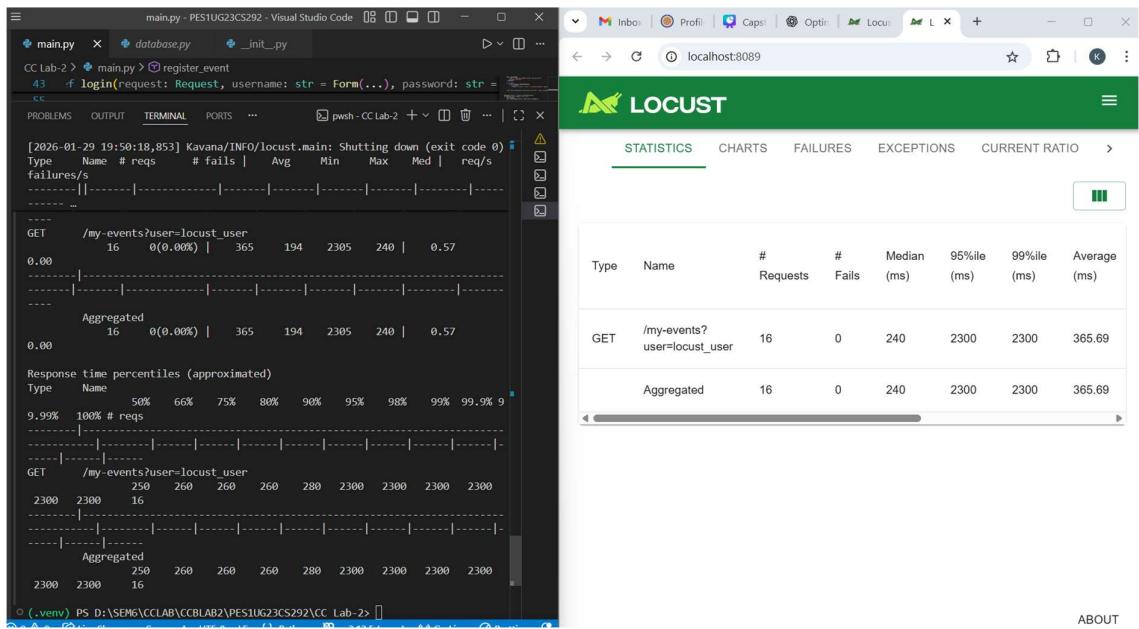
- STATISTICS (selected)
- CHARTS
- FAILURES
- EXCEPTIONS
- CURRENT RATIO
- DOWNLOAD DATA
- LOGS

Table Data:

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	20	0	9	2100	2100	114.84	9	2106	21138	0.7	0
Aggregated		20	0	9	2100	2100	114.84	9	2106	21138	0.7	0

SS8(my_events initial file):

LOCUST		Host http://localhost:8000	Status CLEANUP	RPS 0.6	Failures 0%	EDIT	STOP	RESET	⚙️			
STATISTICS		FAILURES	EXCEPTIONS	CURRENT RATIO	DOWNLOAD DATA	LOGS						
Aggregated												
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	/my-events? user=locust_user	16	0	240	2300	2300	365.69	194	2305	3144	0.6	0
Aggregated		16	0	240	2300	2300	365.69	194	2305	3144	0.6	0



LOCUST		Host http://localhost:8000	Status CLEANUP	RPS 0.6	Failures 0%	EDIT	LOADING	RESET	⚙️			
STATISTICS		FAILURES	EXCEPTIONS	CURRENT RATIO	DOWNLOAD DATA	LOGS						
Aggregated												
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	/my-events? user=locust_user	16	0	240	2300	2300	365.69	194	2305	3144	0.6	0
Aggregated		16	0	240	2300	2300	365.69	194	2305	3144	0.6	0

SS9(MY EVENTS OPTIMIZED):

Bottleneck:

A loop with 1,500,000 iterations was unnecessarily consuming CPU resources for every request.

Change made:

Removed the loop since it did not contribute to the result of the route.

Why performance improved:

By eliminating the artificial computation, the route now responds faster and can handle more concurrent requests.

The screenshot displays the Locust web interface and a terminal window side-by-side.

Locust Web Interface:

- Header:** Host http://localhost:8000, Status RUNNING, Users 1, RPS 0.6, Failures 0%.
- Menu Bar:** STATISTICS, CHARTS, FAILURES, EXCEPTIONS, CURRENT RATIO, DOWNLOAD DATA, LOGS.
- Table Data:**

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	/my-events?user=locust_user	5	0	9	2100	2100	425.78	8	2092	3144	0.6	0
Aggregated		5	0	9	2100	2100	425.78	8	2092	3144	0.6	0
- Terminal Output:** Shows command-line logs for a Python script named main.py running in a Visual Studio Code terminal. It includes logs from the Locust framework and Python code related to event processing and user simulation.
- Browser View:** A separate browser tab shows the Locust dashboard with identical data to the interface above.

Locust Performance Test Results											Actions		
LOCUST		Host		Status		RPS	Failures	Edit		Loading		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	/my-events?user=locust_user	21	0	9	10	2100	107.73	4	2092	3144	0.7	0	
Aggregated		21	0	9	10	2100	107.73	4	2092	3144	0.7	0	

1. Checkout Endpoint (/checkout) – SS5

Performance Issue: The total fee was being calculated using an unnecessarily slow iterative approach.

Modification Done: Replaced the manual accumulation logic with a more efficient built-in summation method.

Result: The endpoint now responds quicker while maintaining the same ability to handle incoming requests.

2. Events Endpoint (/events) – SS7

Performance Issue: A large loop performing redundant calculations was increasing the response time.

Modification Done: Eliminated the loop since it had no impact on the actual output.

Result: The page loads significantly faster and CPU usage during requests is reduced.

3. My Events Endpoint (/my-events) – SS9

Performance Issue: An artificial loop was present that consumed processing time without contributing to functionality.

Modification Done: Removed this unnecessary computation from the route.

Result: The route now executes more efficiently and returns results with improved speed.