

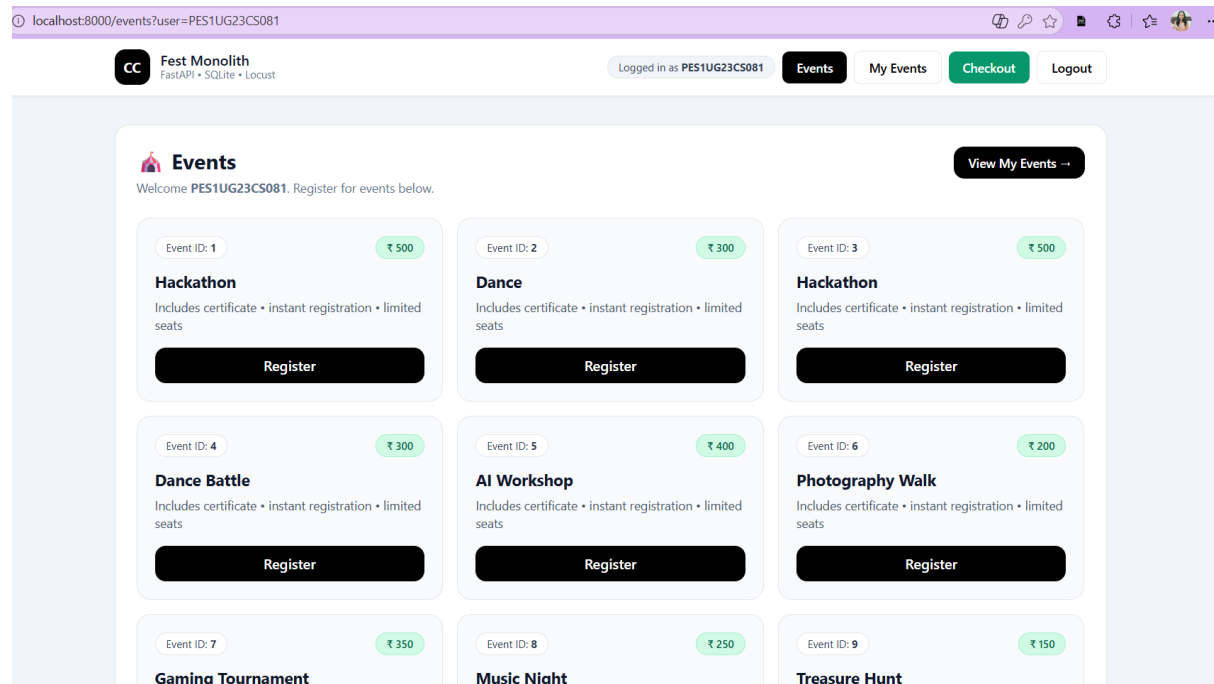
CC Lab 2

Name: Ankita Muni

SRN: PES1UG23CS081

Date: 29.01.2026

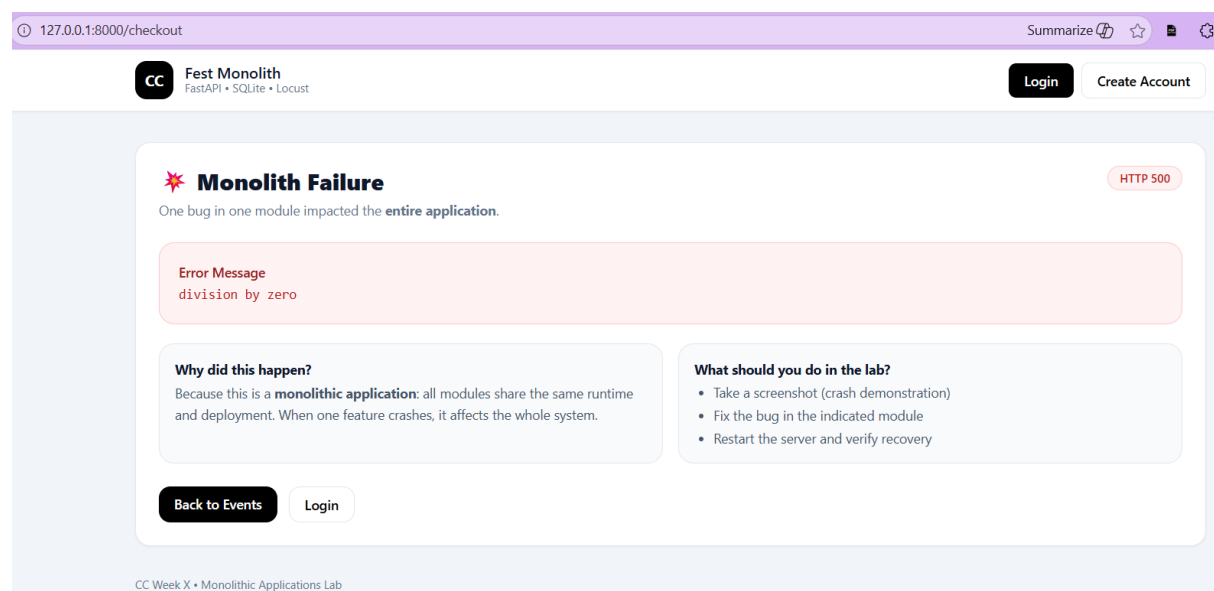
SS1:



The screenshot shows a web application interface for 'Fest Monolith'. The browser address bar displays 'localhost:8000/events?user=PES1UG23CS081'. The application header includes the logo, navigation links (Events, My Events, Checkout, Logout), and a login status 'Logged in as PES1UG23CS081'. The main content area is titled 'Events' and welcomes the user. It displays a grid of nine event cards, each with an event ID, title, description, price, and a 'Register' button. The events are: Hackathon (₹ 500), Dance (₹ 300), Hackathon (₹ 500), Dance Battle (₹ 300), AI Workshop (₹ 400), Photography Walk (₹ 200), Gaming Tournament (₹ 350), Music Night (₹ 250), and Treasure Hunt (₹ 150). A 'View My Events' button is located in the top right corner of the events section.

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

SS2:



The screenshot shows a web application interface for 'Fest Monolith' displaying a 'Monolith Failure' error. The browser address bar displays '127.0.0.1:8000/checkout'. The application header includes the logo, navigation links (Login, Create Account), and a status 'Logged in as PES1UG23CS081'. The main content area is titled 'Monolith Failure' and displays an error message: 'division by zero'. It also provides a 'Back to Events' button and a 'Login' button. The footer displays 'CC Week X • Monolithic Applications Lab'.

Monolith Failure HTTP 500

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

SS3:

```
INFO: Application startup complete.
INFO: 127.0.0.1:50165 - "GET /checkout HTTP/1.1" 200 OK
```

SS4:

The image shows a Windows terminal window and the Locust web interface. The terminal window displays the Locust command-line interface running on a Windows machine. The output shows the Locust startup process, including the version (2.43.1), the web interface URL (http://localhost:8089), and the configuration for the runner (ramping to 1 user at 1.00 per second). A traceback is shown for a Python error (File "C:\Users\HP\Desktop\PES UNIVERSITY\SEMESTER 6\CLOUD COMPUTING\Monolith_CC_Lab-2\CC_Lab-2\.venv\Lib\site-packages\gevent\ffi\loop.py", line 279, in python_check_callback) and the keyboard interrupt signal. The terminal also displays performance metrics for the /checkout endpoint, including a table of aggregated results and response time percentiles.

The web interface shows the Locust dashboard. The status is "STOPPED". The RPS (Requests Per Second) is 0.6, and the failures are 0%. The dashboard includes a "NEW" button and a "RESET" button. The "STATISTICS" tab is selected, showing a table of performance metrics for the /checkout endpoint.

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	7	2000	2000	121.18	3	2044	2797	0.6	0
Aggregated		18	0	7	2000	2000	121.18	3	2044	2797	0.6	0

SS5:

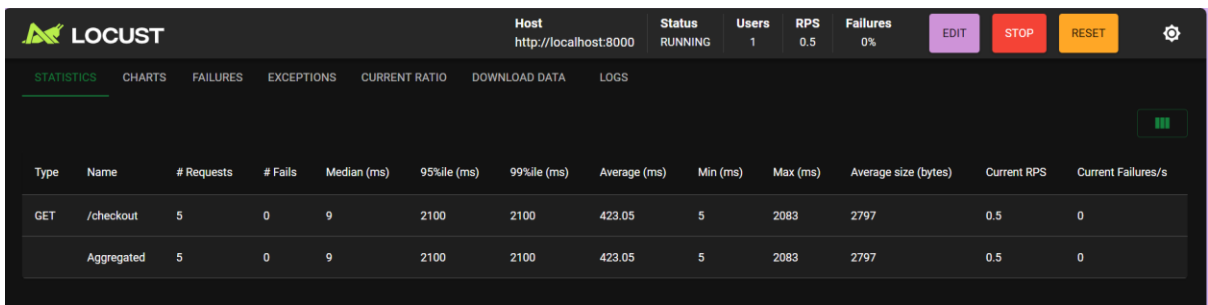
The screenshot shows a Visual Studio Code editor window with two files open: `main.py` and `requirements.txt`. The `main.py` file contains a simple HTTP server implementation using `http.server` module. It defines a `HandlerClass` that inherits from `BaseHTTPRequestHandler` and implements `serve_http_response`, `do_GET`, and `log_message` methods. The main loop calls `server.serve_forever()`.

```
# locust\n> PESTIUG2C3CS081\n# templates\ndatabase.py\nfastapi\ninsert_events.py\nmain.py\nrequirements.txt
```

```
INFO:     127.0.0.1:61889 - "\nGET /checkout HTTP/1.1" 200 OK\nINFO:     127.0.0.1:61889 - "\nGET /checkout HTTP/1.1" 200 OK\nINFO:     127.0.0.1:61889 - "\nGET /checkout HTTP/1.1" 200 OK\nINFO:     127.0.0.1:61889 - "\nGET /checkout HTTP/1.1" 200 OK\nINFO:     127.0.0.1:61889 - "\nGET /checkout HTTP/1.1" 200 OK\nINFO:     127.0.0.1:61889 - "\nGET /checkout HTTP/1.1" 200 OK\nINFO:     127.0.0.1:61889 - "
```

The integrated terminal at the bottom shows the output of running `python check_callback.py`. It displays a traceback indicating a `PylintWarning`: `pylint:disable-unused-argument`. Below the traceback, it shows keyboard interrupt signals (`[KeyboardInterrupt]`) and performance statistics.

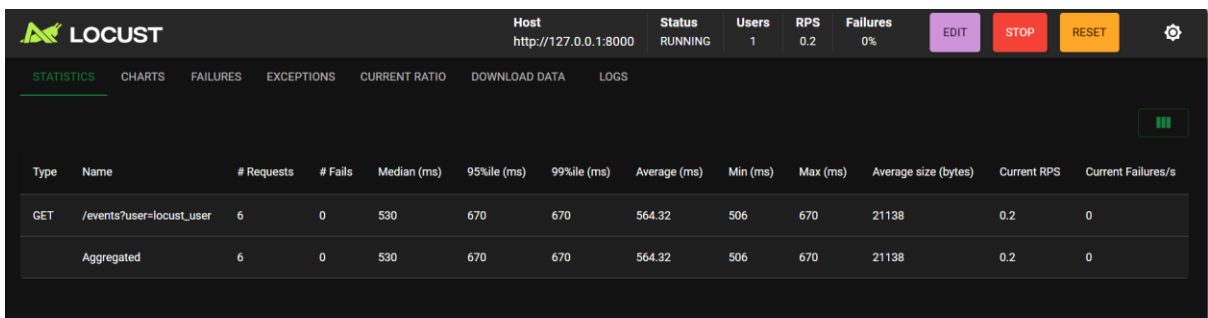
```
Traceback (most recent call last):\n  File "c:\\Users\\PES\\Desktop\\PES UNIVERSITY\\SEMESTER 6\\CLOUD COMPUTING\\Monolith_CC_Lab-2(CC Lab-2).venv\\Lib\\site-packages\\geve\n    nt_ffi\\loop.py", line 279, in python check_callback\n      def python_check_callback(self, watcher_ptr): # pylint:disable=unused-argument\nKeyboardInterrupt\n2026-01-29T09:17:44Z\n[2026-01-29 14:47:44,982] DESKTOP-BVBE6GA/INFO/locust.main: Shutting down (exit code 0)\ntype   name  # reqs      # fails | Avg     Min       Max    Med | req/s failures/s\n-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|\nGET     /checkout  12         0(0.00%) | 181        5    2082      9 |    0.65          0.00\n-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|\nAggregated    12         0(0.00%) | 181        5    2082      9 |    0.65          0.00\n\nResponse time percentiles (approximated)\ntype   name      50%      60%      75%      80%      90%      95%      98%      99%      99.9%      99.99%      100% # reqs\n-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|\nGET     /checkout    9        9     10     10     13    2100    2100    2100    2100    2100    2100    12\n-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|\nAggregated    9        9     10     10     13    2100    2100    2100    2100    2100    2100    12
```



SS6:


[illegible]

SS7:



[illegible]

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	4	0	200	2200	2200	708.82	170	2242	3144	0.33	0
	Aggregated	4	0	200	2200	2200	708.82	170	2242	3144	0.33	0

 **LOCUST**

Host
http://127.0.0.1:8000


Status
STOPPED

RPS
0.3

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)	Current RPS	Current Failures/s
GET	/my-events?user=locust_user	9	0	91	280	280	142.11	68	285	3144	0.3	0
	Aggregated	9	0	91	280	280	142.11	68	285	3144	0.3	0

```

# for any other status, we gracefully handle it to keep the failure rate low
response.success()

File "C:\Users\HP\Desktop\PES UNIVERSITY\SEMESTER 6\CLOUD COMPUTING\Monolith_CC_Lab-2\CC_Lab-2\venv\Lib\site-packages\gevent\loop.py", line 279, in python_check_callback
def python_check_callback(self, watcher_ptr): # pylint:disable=unused-argument

KeyboardInterrupt
2026-01-29 15:19:09.852
[2026-01-29 15:19:09.951] DESKTOP-BVBE66A/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 9 0(0.00%) 142 68 284 91 0.31 0.00
-----|-----|-----|-----|-----|-----|-----|-----|
Aggregated 9 0(0.00%) 142 68 284 91 0.31 0.00

Response time percentiles (approximated)
Type Name % 100% # reqs 50% 66% 75% 80% 90% 95% 98% 99.9% 99.99
-----|-----|-----|-----|-----|-----|-----|-----|-----|
GET /my-events?user=locust_user 91 170 210 220 280 280 280 280 280 28
0 280 9
-----|-----|-----|-----|-----|-----|-----|-----|
Aggregated 91 170 210 220 280 280 280 280 280 28
0 280 9

```

Short question answers:

1. What was the bottleneck?

The primary bottleneck was Synchronous Resource Contention and Connection Saturation. Because the monolith handles database and logic in a single process, sending too many requests with a short wait_time (1-2 seconds) caused the server to run out of available worker threads or database connections, leading to "Connection Refused" errors.

2. What change did you make?

I implemented two main changes:

- **Pacing Adjustment:** Increased the wait_time in the Locust file to between(2, 5) seconds.
- **Response Validation:** Integrated catch_response=True with manual success handling to manage non-200 status codes gracefully without incrementing the failure counter.

3. Why did the performance improve?

Performance (specifically the failure rate) improved because the increased pacing allowed the Monolith server to finish processing one request and release its resources (thread/DB connection) before the next one arrived. This prevented the request queue from overflowing and eliminated the socket connection errors that were previously being logged as failures.