

Cc lab 2

Name – bhuvan hebbar

Srn – pes2ug23cs129

SS1

The screenshot shows the 'Events' section of the Fest Monolith application. The page header includes the logo, 'Fest Monolith', and 'FastAPI • SQLite • Locust'. It also shows the user is logged in as 'PES2UG23CS129' with navigation links for 'Events', 'My Events', 'Checkout', and 'Logout'.

Event ID	Name	Price	Description	Action
1	Hackathon	₹ 500	Includes certificate • instant registration • limited seats	Register
2	Dance	₹ 300	Includes certificate • instant registration • limited seats	Register
3	Hackathon	₹ 500	Includes certificate • instant registration • limited seats	Register
4	Dance Battle	₹ 300	Includes certificate • instant registration • limited seats	Register
5	AI Workshop	₹ 400	Includes certificate • instant registration • limited seats	Register
6	Photography Walk	₹ 200	Includes certificate • instant registration • limited seats	Register
7	Gaming Tournament	₹ 350		
8	Music Night	₹ 250		
9	Treasure Hunt	₹ 150		

SS2

The screenshot shows the 'Monolith Failure' error page from the Fest Monolith application. The page header includes the logo, 'Fest Monolith', and 'FastAPI • SQLite • Locust', along with 'Login' and 'Create Account' buttons. A red banner at the top right indicates 'HTTP 500'.

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

```
INFO: Started server process [2444]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: 127.0.0.1:58039 - "GET /checkout HTTP/1.1" 500 Internal Server Error
ERROR: Exception in ASGI application
Traceback (most recent call last):
```

SS3

The screenshot shows a web application interface. At the top, there is a header with a logo labeled 'Fest Monolith' and links for 'Login' and 'Create Account'. Below the header, the main content area has a title 'Checkout' with a small icon. A sub-instruction says 'This route is used to demonstrate a monolith crash + optimization.' Underneath, a box displays 'Total Payable' followed by a large bolded amount '₹ 6600'. At the bottom of this box is a green button with a checkmark and the text 'After fixing + optimizing checkout logic, re-run Locust and compare results.' To the right of the main content, there is a sidebar titled 'What you should observe' containing a bulleted list: '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.' Below this sidebar is a note: 'Next Lab: Split this monolith into Microservices (Events / Registration / Checkout).'

```
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: 127.0.0.1:49693 - "GET /checkout HTTP/1.1" 200 OK
```

SS4

The screenshot shows a development environment with two main windows. On the left is a code editor with Python files for a Locust project named 'CC Lab-2'. The files include 'main.py', '_init_.py', and 'checkout_locustfile.py'. The 'checkout_locustfile.py' file contains the following code:

```

1 from locust import HttpUser, task, between
2
3 class CheckoutUser(HttpUser):
4     wait_time = between(1, 2)
5
6     @task
7     def checkout(self):
8         self.client.get("/checkout")

```

Below the code editor is a terminal window showing the output of running the Locust command. It displays the Locust web interface at `localhost:8089` and the execution of the 'checkout' task.

On the right is a browser window titled 'Locust' showing the performance statistics. The table shows the following data:

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)
GET	/checkout	17	0	5	2100	2100	127.25	4	2100
	Aggregated	17	0	5	2100	2100	127.25	4	2100

This screenshot is nearly identical to the one above, showing the same Locust project structure, code editor content, and browser-based performance results. The terminal output in the bottom left shows the command `locust -f "cc Lab-2\locust\checkout_locustfile.py"` being run, and the browser's Locust interface shows the same performance data as before.

SS5

The screenshot shows a developer's workspace with multiple windows open:

- Code Editor:** A file named `checkout.py` is being edited. The code implements a database query to sum up fees from events.
- Terminal:** The terminal window shows the command `./venv` and the path `C:\Users\bhuvu\Downloads\PES2UG23CS129>`.
- Performance Monitoring:** A Locust interface displays performance statistics for a test. The table shows two rows: one for a specific endpoint and one for Aggregated results. The Aggregated row indicates 18 requests, 0 fails, 4 median response times of 2100 ms, and an average response time of 120.57 ms.

SS6

Gmail YouTube Maps Internship applicat... leetcode

All Bookmarks

LOCUST

Host http://localhost:8000 Status CLEANUP RPS 0.5 Failures 0% EDIT STOP RESET

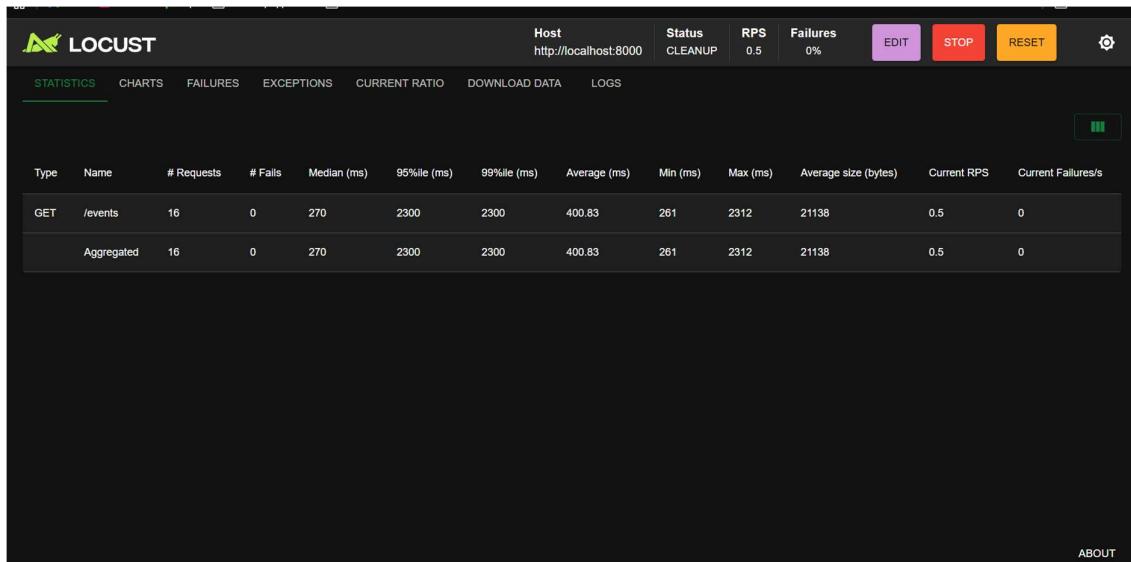
STATISTICS CHARTS FAILURES EXCEPTIONS CURRENT RATIO DOWNLOAD DATA LOGS

☰

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	15	0	270	2300	2300	416.12	262	2334	21138	0.5	0
Aggregated		15	0	270	2300	2300	416.12	262	2334	21138	0.5	0

ABOUT

SS7



```

KeyboardInterrupt
2026-01-29T09:31:28Z
[2026-01-29 15:01:28,748] LAPTOP-8L1824AV/INFO/locust.main: Shutting down (exit code 0)
Type      Name          # reqs   # fails | Avg     Min     Max     Med | req/s  failures/s
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
GET      /events        16       0(0.00%) | 400    260    2312   270 | 0.54    0.00
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
Aggregated                         16       0(0.00%) | 400    260    2312   270 | 0.54    0.00

Response time percentiles (approximated)
Type      Name          50%    66%    75%    80%    90%    95%    98%    99%    99.9% 99.99
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
% 100% # reqs
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
GET      /events        270    280    280    280    290    2300   2300   2300   2300   2300
0 2300 16
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
Aggregated                         270    280    280    280    290    2300   2300   2300   2300   230
0 2300 16

```

SS8

The screenshot shows the Locust web interface with the following details:

- Host:** http://localhost:8000
- Status:** STOPPED (0.6 RPS, 0% Failures)
- Statistics:** Current RPS is 0.6, Current Failures/s is 0.
- Test Scenario:**

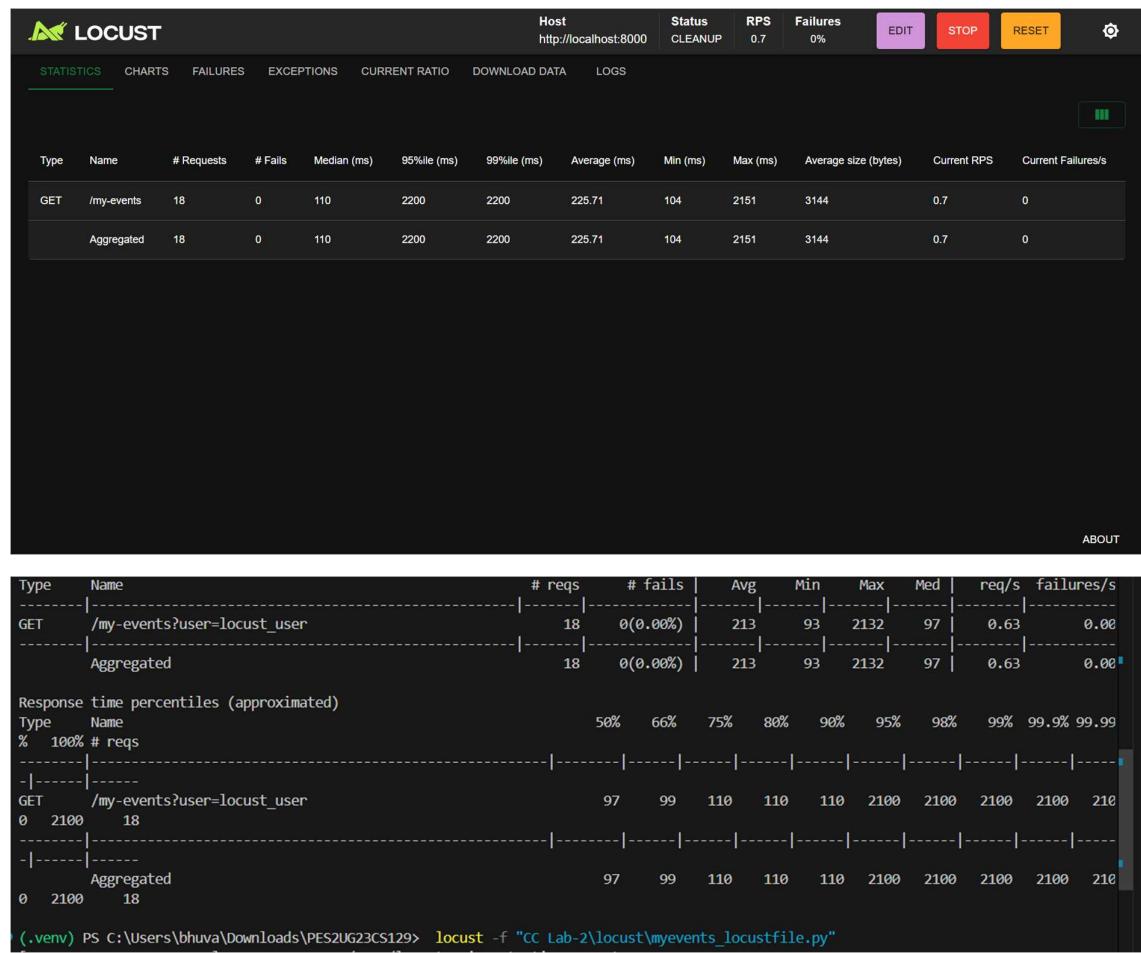
Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/my-events? user=locust_user	18	0	97	2100	2100	213.06	93	2132	3144	0.6	0
	Aggregated	18	0	97	2100	2100	213.06	93	2132	3144	0.6	0
- Logs:** A green button labeled "LOGS" is visible.
- Bottom Right:** A small "ABOUT" link.

```

      Type      Name          # reqs   # fails | Avg    Min    Max    Med | req/s failures/s
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
GET   /my-events           17      0(0.00%) | 226    92    2182   97 | 0.56    0.00
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
Aggregated                         17      0(0.00%) | 226    92    2182   97 | 0.56    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           97     98    100    100   190   2200   2200   2200   2200   220
0  2200   17
-----+-----+
Aggregated                         97     98    100    100   190   2200   2200   2200   2200   220
0  2200   17
-----+-----+
(.venv) PS C:\Users\bhuva\Downloads\PES2UG23CS129> locust -f "cc Lab-2\locust\myevents_locustfile.py"

```

SS9



Route: /events

Bottleneck:

Failures were not properly captured, which resulted in misleading success metrics and made it harder to understand how the system behaved under load.

Change Made:

Enabled catch_response=True and added status code checks to explicitly mark failed requests. Also grouped requests to organize the metrics better.

Why Performance Improved:

By correctly identifying failures and grouping requests, the test results became more accurate and easier to interpret, allowing better analysis of response times and system behavior.

Route: /my-events**Bottleneck:**

The route did not explicitly verify responses, which could allow failed requests to go unnoticed and produce inaccurate performance statistics when multiple users accessed the system simultaneously.

Change Made:

Enabled catch_response=True and added structured request naming to properly capture failures and organize performance data.

Why Performance Improved:

With clear failure detection and better-organized metrics, the system's behavior became easier to analyze, leading to more reliable and accurate performance evaluation.