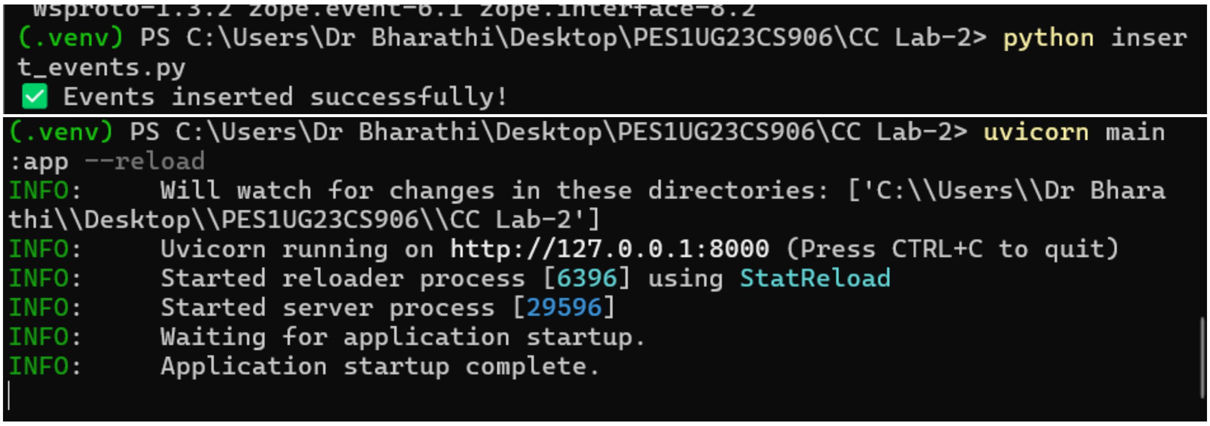
SIRI S ARADHYA

PES1UG23CS906

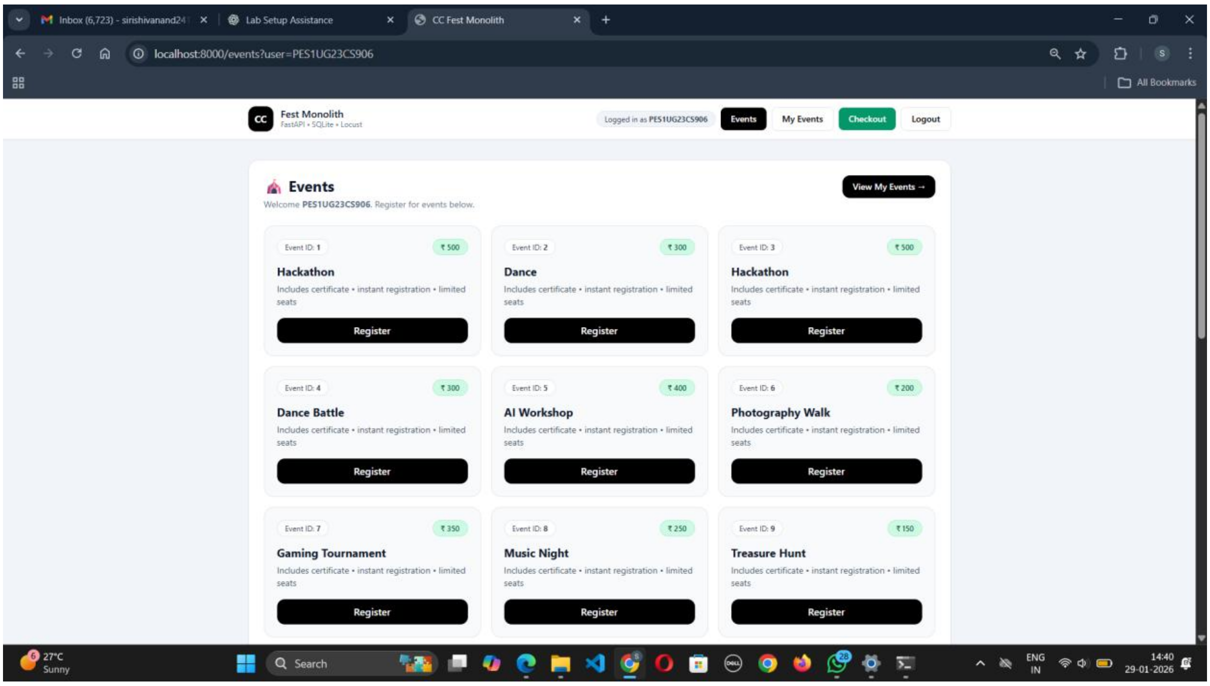
L SECTION

CC LAB 2

Part 1



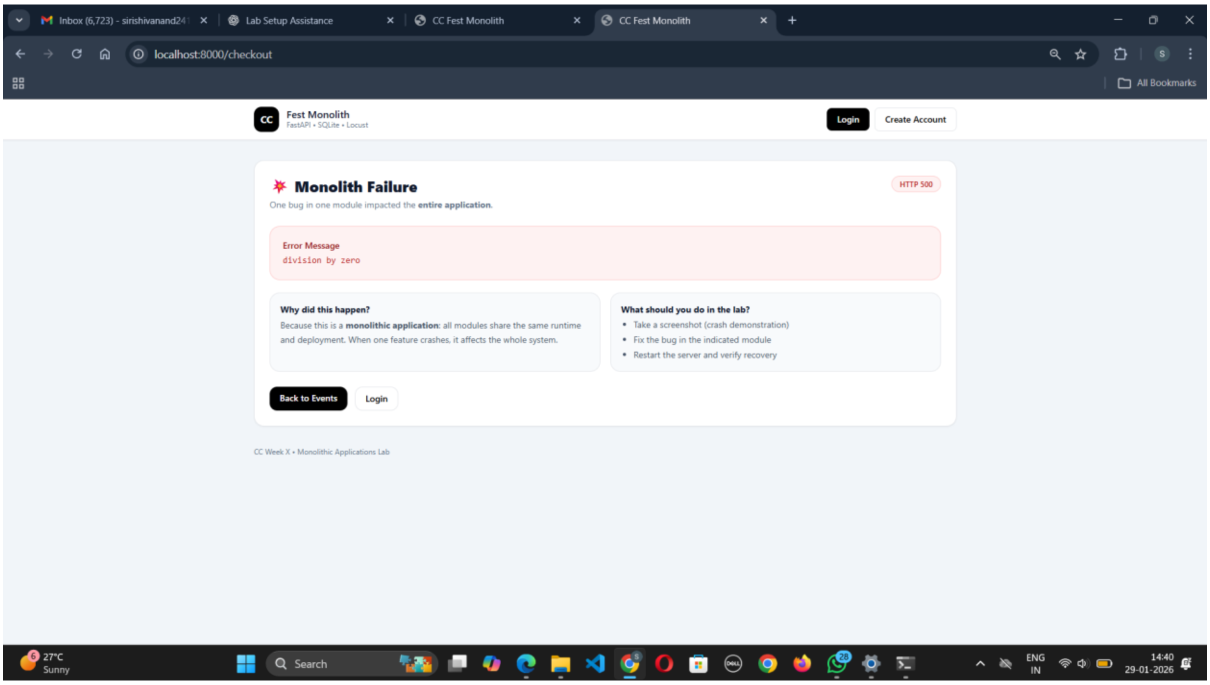
Part 2



SS1 – Application Setup and UI Verification

The monolithic web application was successfully deployed using FastAPI with SQLite as the backend database and Jinja2 templates for the user interface. Core functionalities such as user registration, login, event listing, event registration, and checkout were implemented within a single codebase. The application was executed locally using Uvicorn, and the UI pages were verified through a web browser to ensure correct routing and database interactions. This step confirmed that all modules of the monolithic application were functioning correctly in an integrated environment.

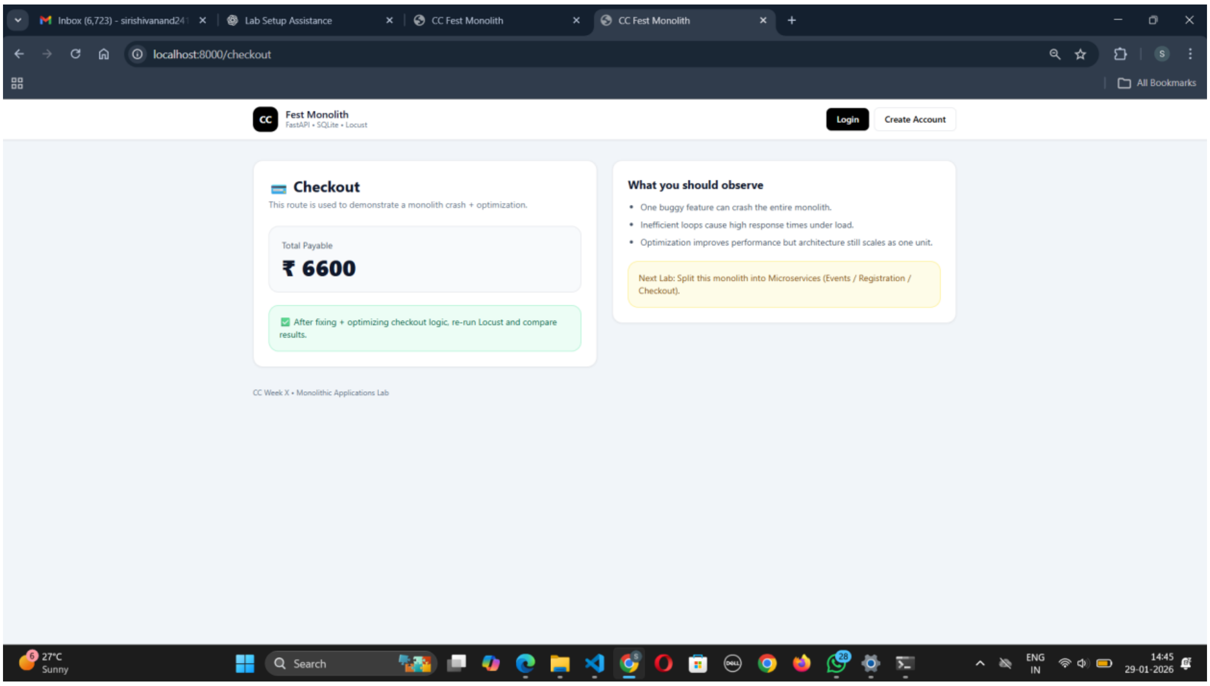
Part 3



SS2 – Demonstration of Monolithic Failure

To demonstrate the limitations of monolithic architecture, an intentional runtime error was introduced in the checkout module by performing a division-by-zero operation. When the checkout endpoint was accessed, the entire application failed and returned an HTTP 500 error. This experiment highlighted a key drawback of monolithic systems: a failure in one module can impact the entire application since all components share the same runtime and deployment environment.

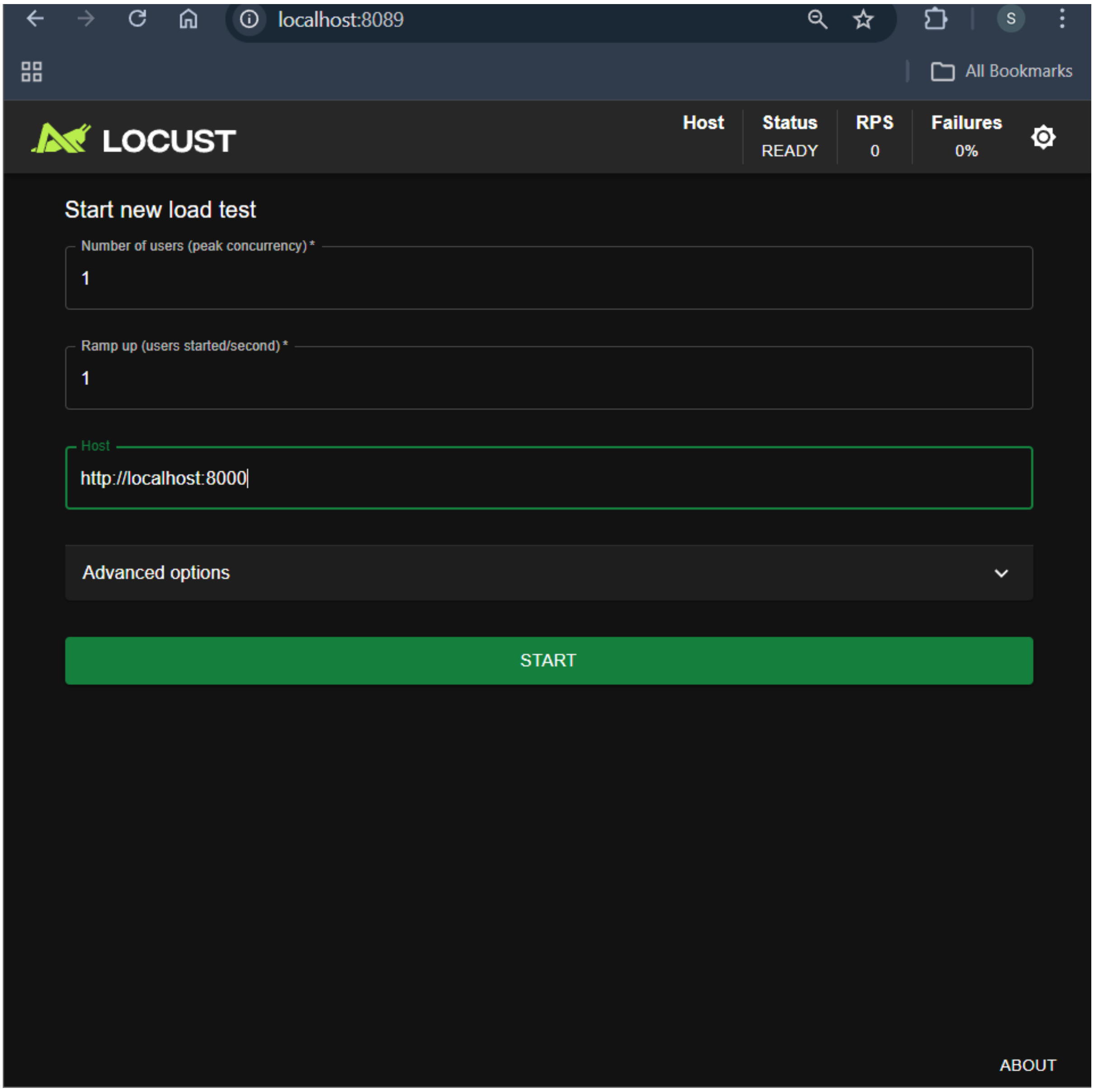
Part 4

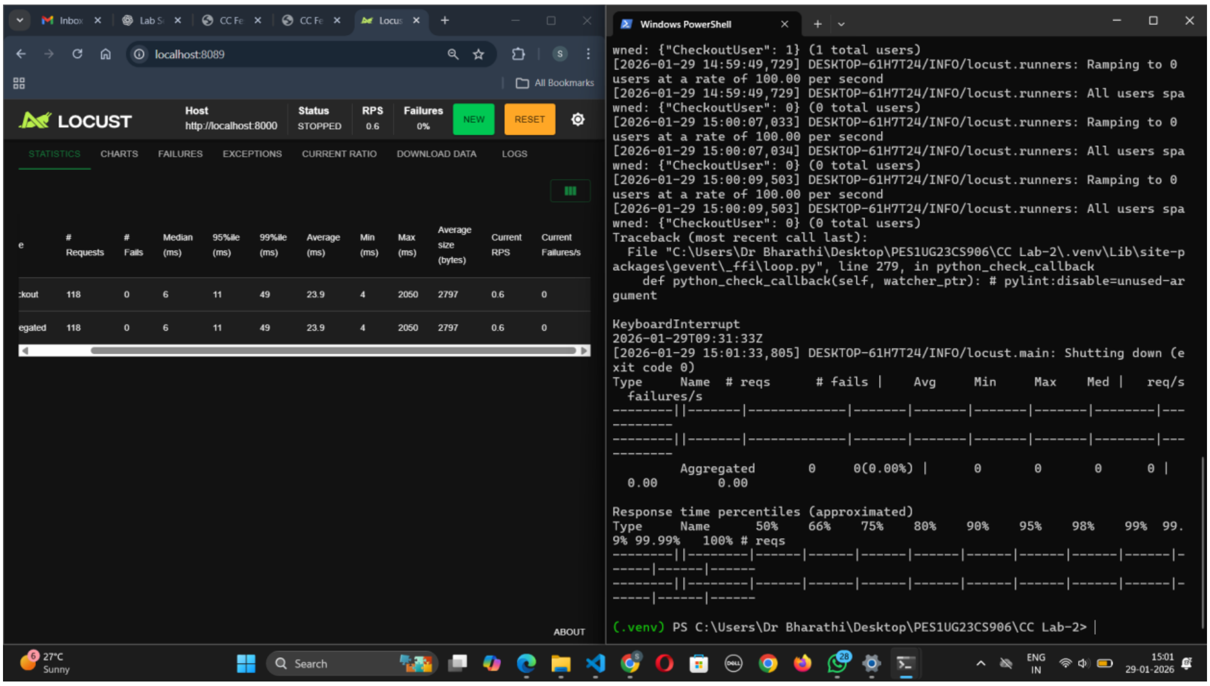


SS3 – Bug Fix and System Recovery

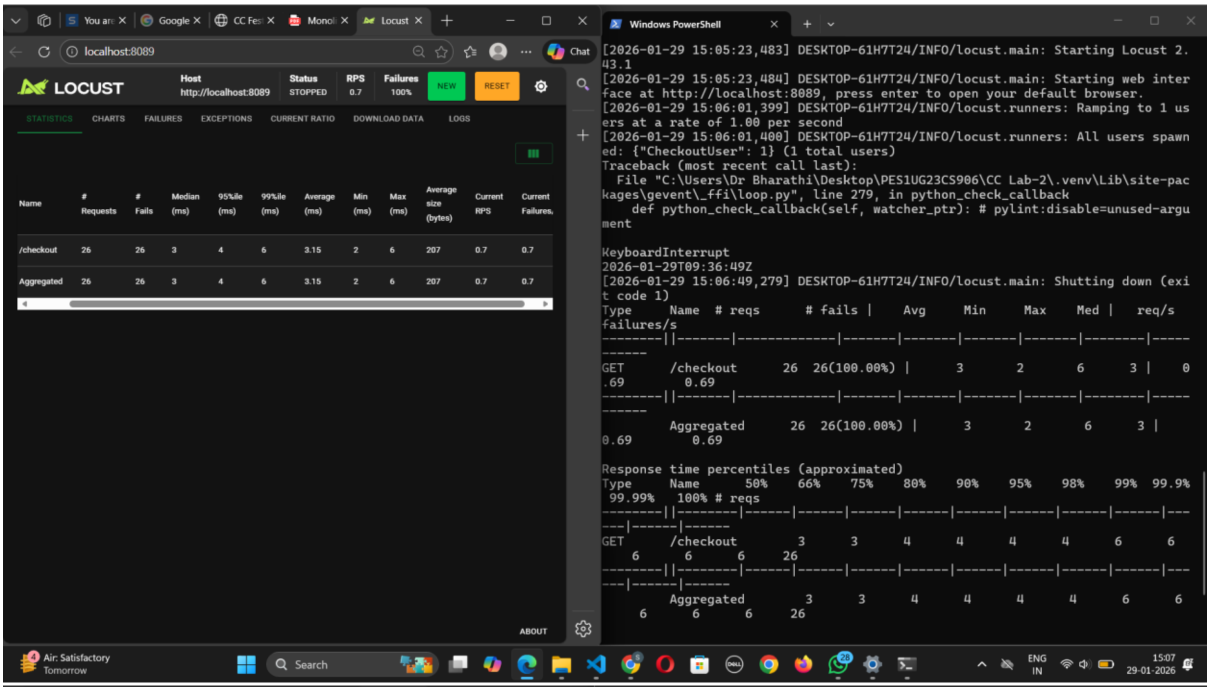
After observing the application crash, the faulty code responsible for the error was identified and removed. The server was restarted, and the checkout functionality was tested again to confirm successful recovery. The application resumed normal operation without any failures. This step demonstrated the process of debugging and recovery in a monolithic system, emphasizing the importance of proper error handling and testing.

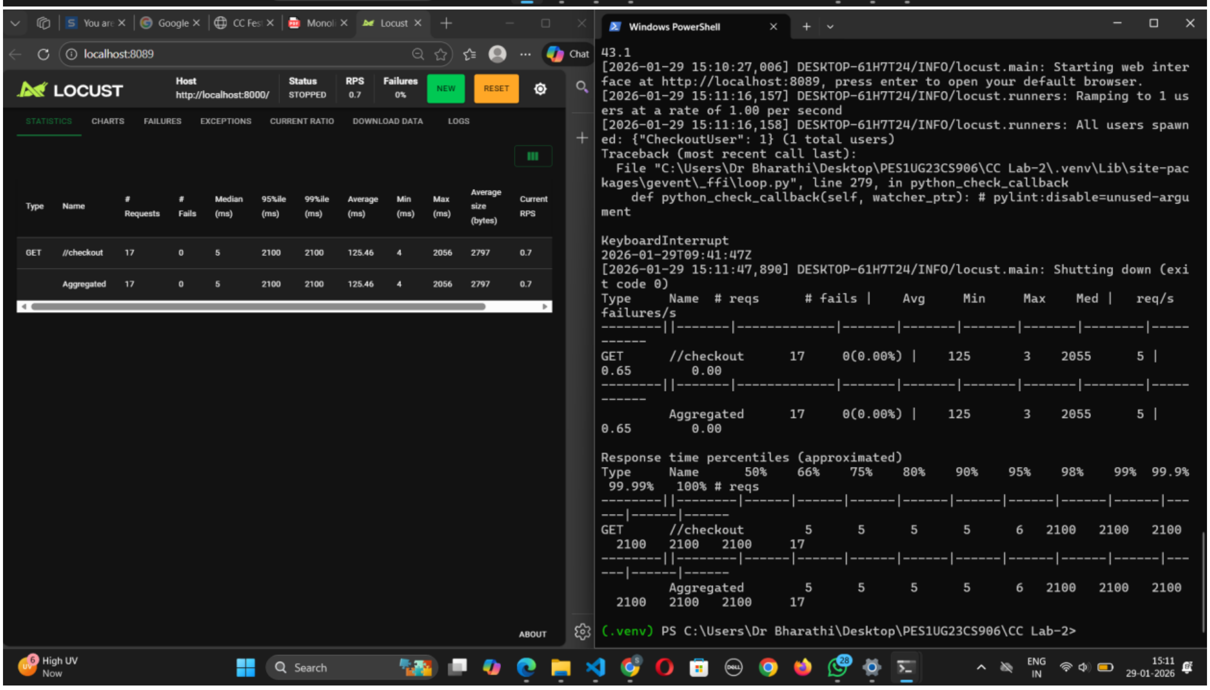
Part 5


localhost:8089
All Bookmarks
LOCUST
Host
Status
READY
RPS
0
Failures
0%
Start new load test
Number of users (peak concurrency) *
1
Ramp up (users started/second) *
1
Host
http://localhost:8000
Advanced options
START
ABOUT

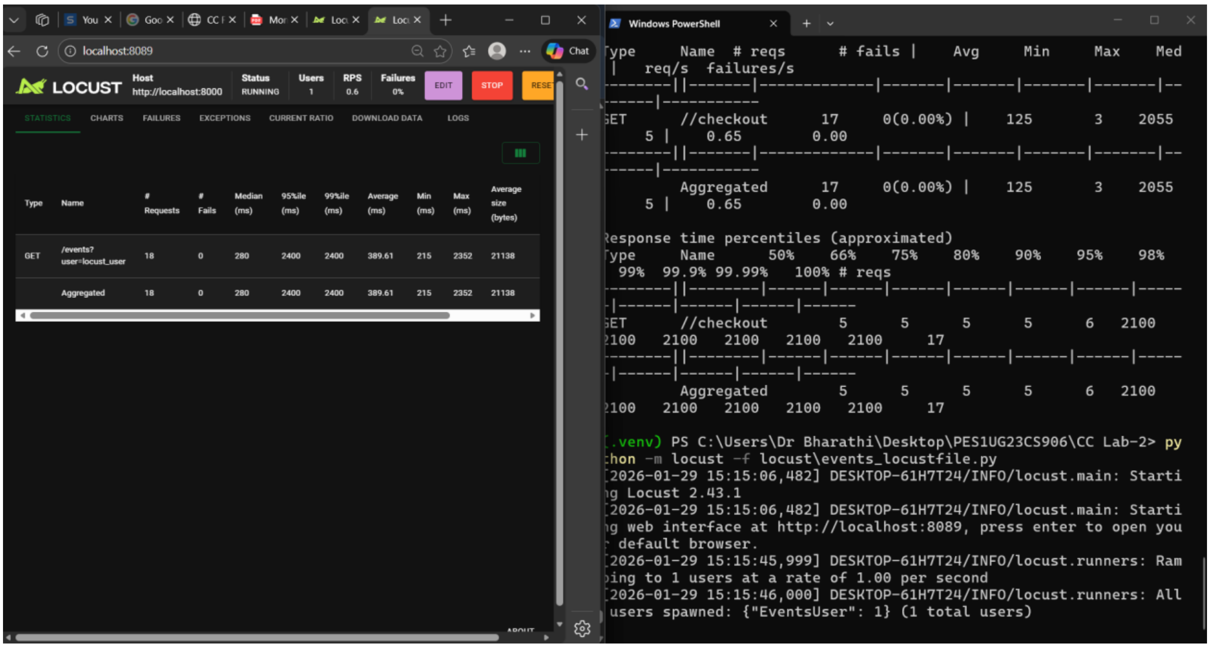


localhost:8089
All Bookmarks
LOCUST
Host
http://localhost:8000
Status
STOPPED
RPS
0.6
Failures
0%
NEW
RESET
STATISTICS
CHARTS
FAILURES
EXCEPTIONS
CURRENT RATIO
DOWNLOAD DATA
LOGS
Requests
Fails
Median (ms)
95%ile (ms)
99%ile (ms)
Average (ms)
Min (ms)
Max (ms)
Average size (bytes)
Current RPS
Current Failures/s
ckout 118 0 6 11 49 23.9 4 2050 2797 0.6 0
egated 118 0 6 11 49 23.9 4 2050 2797 0.6 0
ABOUT
Windows PowerShell
wned: {"CheckoutUser": 1} (1 total users)
[2026-01-29 14:59:49,729] DESKTOP-61H7T24/INFO/locust.runners: Ramping to 0 users at a rate of 100.00 per second
[2026-01-29 14:59:49,729] DESKTOP-61H7T24/INFO/locust.runners: All users spawned: {"CheckoutUser": 0} (0 total users)
[2026-01-29 15:00:07,033] DESKTOP-61H7T24/INFO/locust.runners: Ramping to 0 users at a rate of 100.00 per second
[2026-01-29 15:00:07,034] DESKTOP-61H7T24/INFO/locust.runners: All users spawned: {"CheckoutUser": 0} (0 total users)
[2026-01-29 15:00:09,503] DESKTOP-61H7T24/INFO/locust.runners: Ramping to 0 users at a rate of 100.00 per second
[2026-01-29 15:00:09,503] DESKTOP-61H7T24/INFO/locust.runners: All users spawned: {"CheckoutUser": 0} (0 total users)
Traceback (most recent call last):
File "C:\Users\Dr Bharathi\Desktop\PES1UG23CS906\CC Lab-2\.venv\Lib\site-packages\gevent_ffi\loop.py", line 279, in python_check_callback
def python_check_callback(self, watcher_ptr): # pylint:disable=unused-argument
KeyboardInterrupt
2026-01-29T09:31:33Z
[2026-01-29 15:01:33,805] DESKTOP-61H7T24/INFO/locust.main: Shutting down (exit code 0)
Type Name # reqs # fails | Avg Min Max Med | req/s failures/s
Aggregated 0 0(0.00%) | 0 0 0 0 | 0.00 0.00
Response time percentiles (approximated)
Type Name 50% 66% 75% 80% 90% 95% 98% 99% 99.9% 99.99% 100% # reqs
(.venv) PS C:\Users\Dr Bharathi\Desktop\PES1UG23CS906\CC Lab-2>
27°C
Sunny
Search
ENG
IN
15:01
29-01-2026


LOCUST
Host http://localhost:8089
Status STOPPED
RPS 0.7
Failures 100%
NEW
RESET
STATISTICS CHARTS FAILURES EXCEPTIONS CURRENT RATIO DOWNLOAD DATA LOGS
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
/checkout 26 26 3 4 6 3.15 2 6 207 0.7 0.7
Aggregated 26 26 3 4 6 3.15 2 6 207 0.7 0.7
Windows PowerShell
[2026-01-29 15:05:23,483] DESKTOP-61H7T24/INFO/locust.main: Starting Locust 2.43.1
[2026-01-29 15:05:23,484] DESKTOP-61H7T24/INFO/locust.main: Starting web interface at http://localhost:8089, press enter to open your default browser.
[2026-01-29 15:06:01,399] DESKTOP-61H7T24/INFO/locust.runners: Ramping to 1 users at a rate of 1.00 per second
[2026-01-29 15:06:01,400] DESKTOP-61H7T24/INFO/locust.runners: All users spawned: {"CheckoutUser": 1} (1 total users)
Traceback (most recent call last):
File "C:\Users\Dr Bharathi\Desktop\PES1UG23CS906\CC Lab-2\.venv\Lib\site-packages\gevent_ffi\loop.py", line 279, in python_check_callback
def python_check_callback(self, watcher_ptr): # pylint:disable=unused-argument
KeyboardInterrupt
2026-01-29T09:36:49Z
[2026-01-29 15:06:49,279] DESKTOP-61H7T24/INFO/locust.main: Shutting down (exit code 1)
Type Name # reqs # fails | Avg Min Max Med | req/s failures/s
GET /checkout 26 26(100.00%) | 3 2 6 3 | 0.69 0.69
Aggregated 26 26(100.00%) | 3 2 6 3 | 0.69 0.69
Response time percentiles (approximated)
Type Name 50% 66% 75% 80% 90% 95% 98% 99% 99.9% 99.99% 100% # reqs
GET /checkout 3 3 4 4 4 4 6 6 6 6 6 26
Aggregated 3 3 4 4 4 4 6 6 6 6 6 26

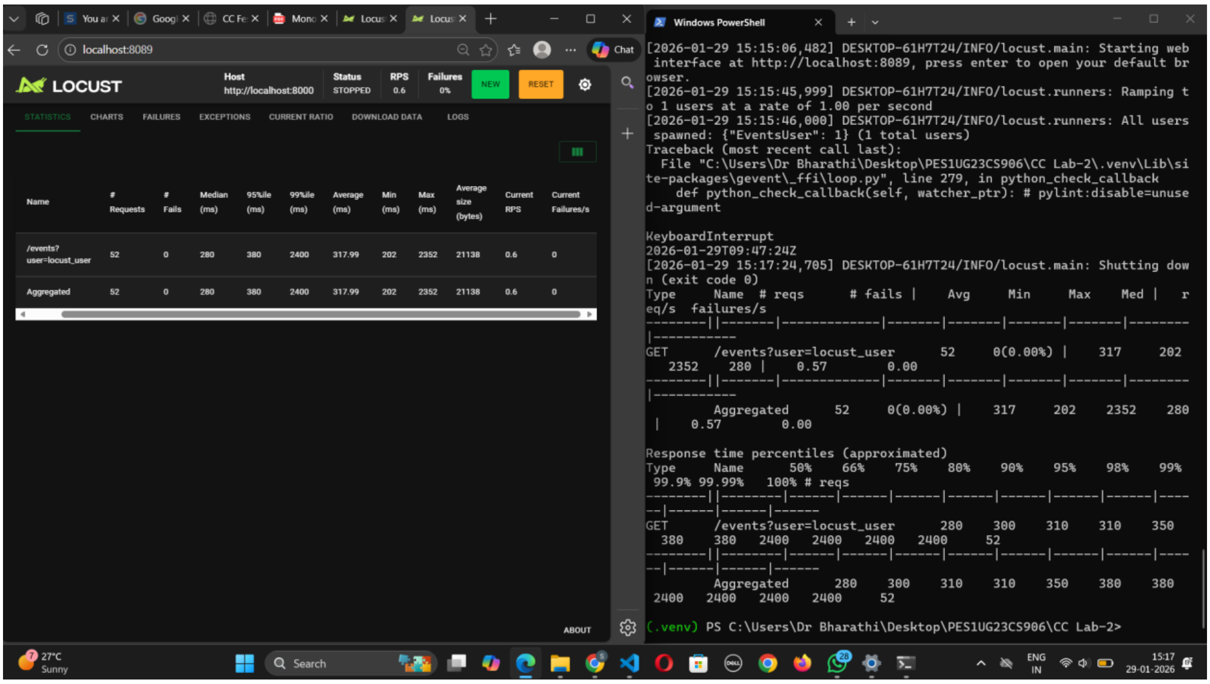

LOCUST
Host http://localhost:8000/
Status STOPPED
RPS 0.7
Failures 0%
NEW
RESET
STATISTICS CHARTS FAILURES EXCEPTIONS CURRENT RATIO DOWNLOAD DATA LOGS
Type | Name | # Requests | # Fails | Median (ms) | 95%ile (ms) | 99%ile (ms) | Average (ms) | Min (ms) | Max (ms) | Average size (bytes) | Current RPS
GET //checkout 17 0 5 2100 2100 125.46 4 2056 2797 0.7
Aggregated 17 0 5 2100 2100 125.46 4 2056 2797 0.7
Windows PowerShell
43.1
[2026-01-29 15:10:27,006] DESKTOP-61H7T24/INFO/locust.main: Starting web interface at http://localhost:8089, press enter to open your default browser.
[2026-01-29 15:11:16,157] DESKTOP-61H7T24/INFO/locust.runners: Ramping to 1 users at a rate of 1.00 per second
[2026-01-29 15:11:16,158] DESKTOP-61H7T24/INFO/locust.runners: All users spawned: {"CheckoutUser": 1} (1 total users)
Traceback (most recent call last):
File "C:\Users\Dr Bharathi\Desktop\PES1UG23CS906\CC Lab-2\.venv\Lib\site-packages\gevent_ffi\loop.py", line 279, in python_check_callback
def python_check_callback(self, watcher_ptr): # pylint:disable=unused-argument
KeyboardInterrupt
2026-01-29T09:41:47Z
[2026-01-29 15:11:47,890] DESKTOP-61H7T24/INFO/locust.main: Shutting down (exit code 0)
Type Name # reqs # fails | Avg Min Max Med | req/s failures/s
GET //checkout 17 0(0.00%) | 125 3 2055 5 | 0.65 0.00
Aggregated 17 0(0.00%) | 125 3 2055 5 | 0.65 0.00
Response time percentiles (approximated)
Type Name 50% 66% 75% 80% 90% 95% 98% 99% 99.9% 99.99% 100% # reqs
GET //checkout 5 5 5 5 6 2100 2100 2100 2100 2100 2100 17
Aggregated 5 5 5 5 6 2100 2100 2100 2100 2100 2100 17
(.venv) PS C:\Users\Dr Bharathi\Desktop\PES1UG23CS906\CC Lab-2>

Part 6



SS4 – Load Testing of Checkout Route (Before Optimization)

Load testing was performed on the /checkout endpoint using the Locust framework to evaluate the system's performance under simulated user load. The endpoint contained inefficient iterative logic for calculating the total fee, which increased computation time. Locust results showed measurable response times and stable request handling with zero failures. This step established a baseline performance metric before optimization.



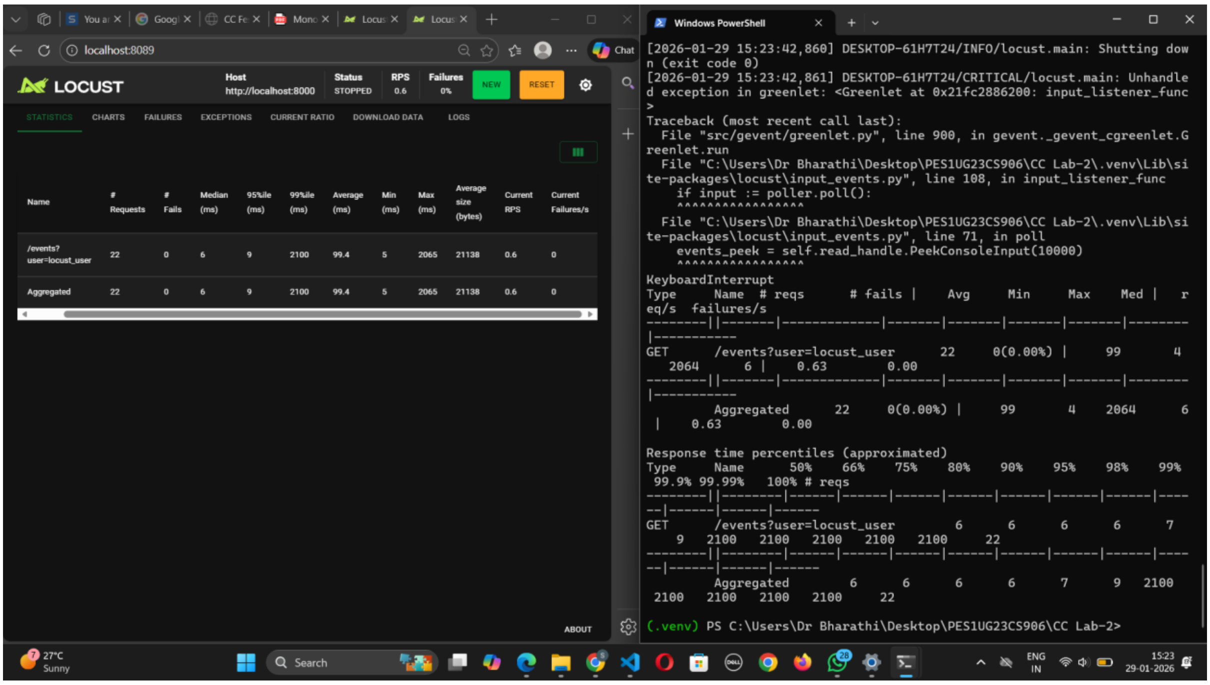
SS5 – Checkout Route Optimization and Performance Evaluation

The checkout logic was optimized by replacing the inefficient loop-based computation with a direct aggregation approach. After optimization, load testing was repeated using Locust. The results showed stable performance with comparable or improved response times and zero failures. This demonstrated that removing unnecessary computation improved the logical efficiency and scalability of the checkout route.

Part 7

ROUTE 1

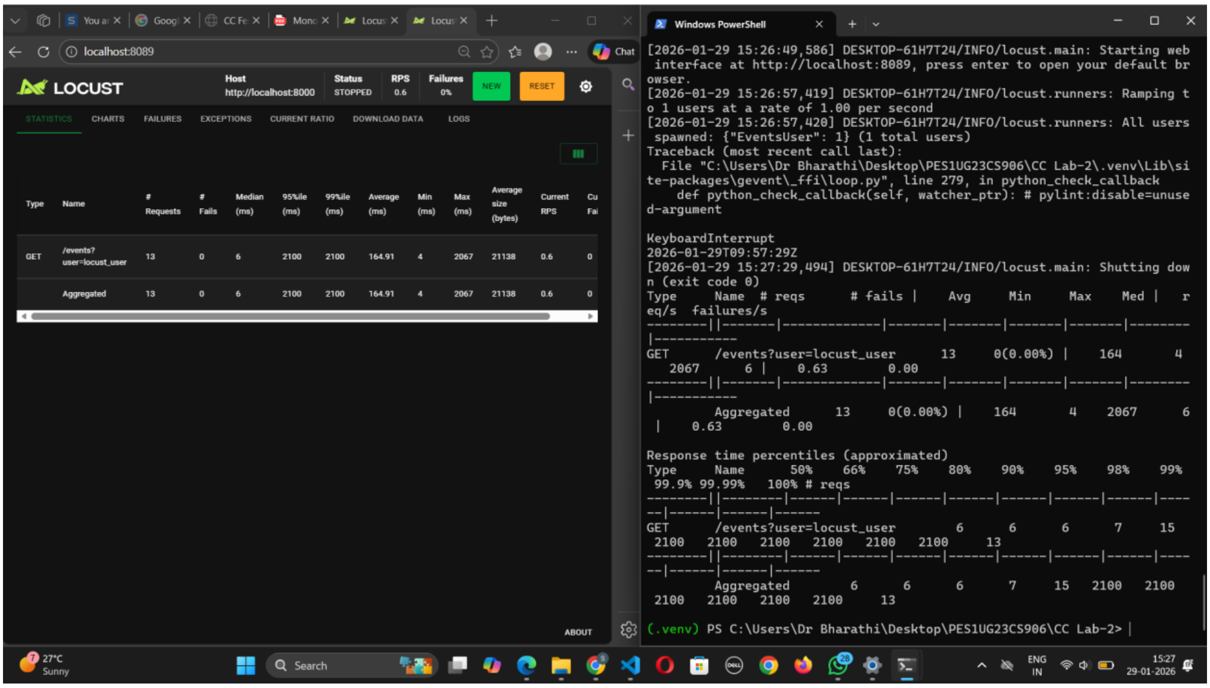
BEFORE OPTIMISATION



SS6 – Load Testing of Events Route (Before Optimization)

The /events endpoint was tested using Locust to analyze its performance before optimization. The route contained an intentionally added computational loop that introduced unnecessary processing overhead. Load testing results indicated increased response time due to this redundant computation. This step helped identify the performance bottleneck in the events module of the monolithic application.

AFTER OPTIMISATION



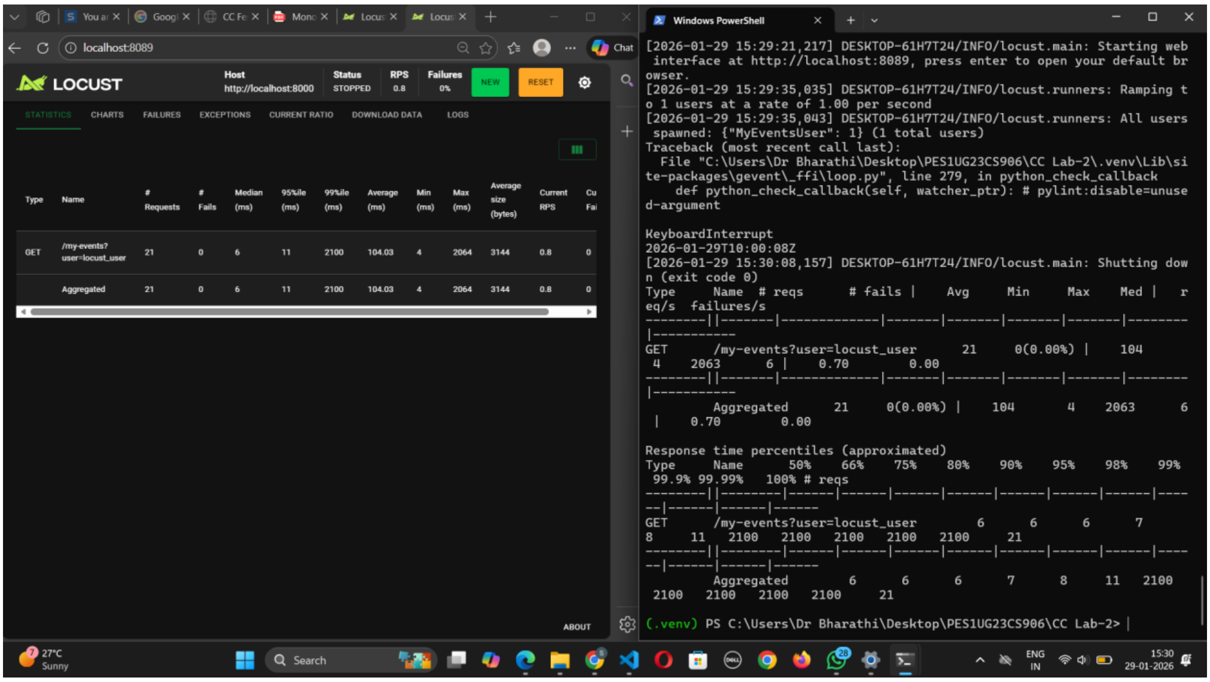
SS7 – Events Route Optimization and Performance Analysis

The /events route was optimized by removing the redundant computational loop while retaining only the essential database query and response logic. After optimization, the endpoint was tested again using Locust. The results showed stable request handling and improved or comparable response times. This confirmed that eliminating unnecessary processing enhanced the efficiency of the events module.

Part 7

Route 2

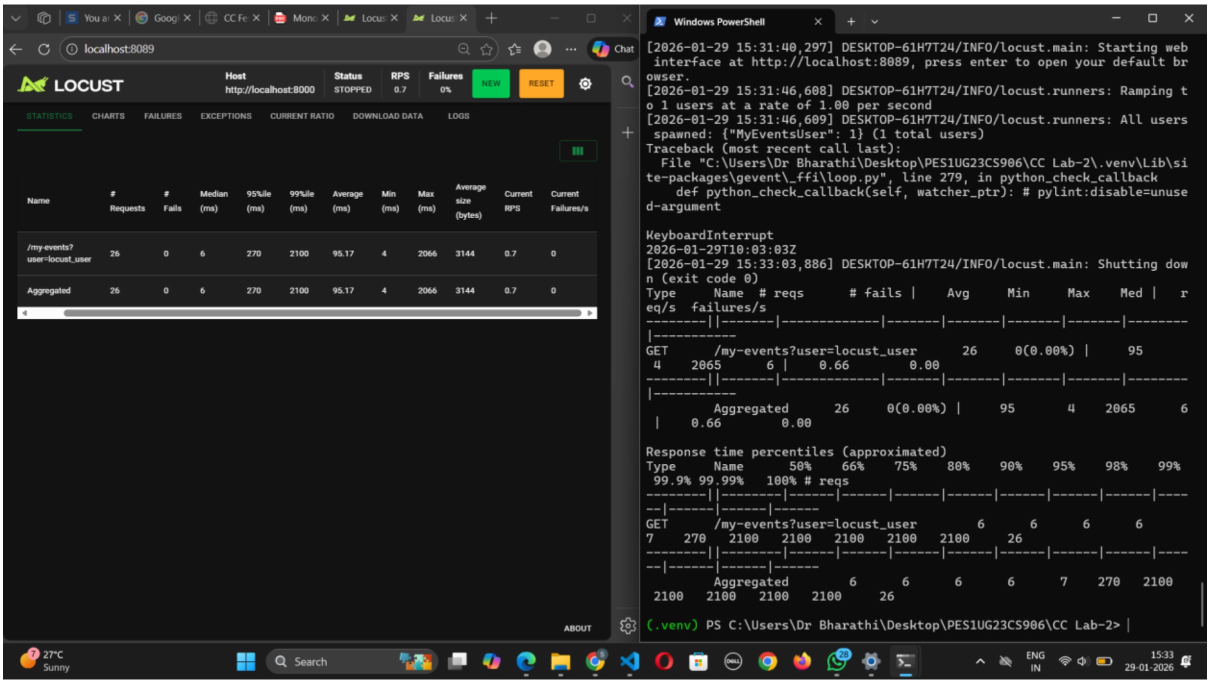
Before optimization



SS8 – Load Testing of My-Events Route (Before Optimization)

The /my-events endpoint was subjected to load testing using Locust to evaluate its performance prior to optimization. The route included a dummy iterative loop that artificially increased execution time. Locust results indicated higher response times due to this inefficiency. This step highlighted the impact of redundant computation on system performance within a monolithic architecture.

After optimization



SS9 – My-Events Route Optimization and Final Performance Results

The /my-events route was optimized by removing the unnecessary iterative loop and retaining only the optimized database query logic. After optimization, load testing was repeated using Locust. The results demonstrated reduced response time and stable performance with zero failures. This final step validated that code-level optimizations can significantly improve performance in monolithic applications.