

CC LAB-2

Name: Laharish S

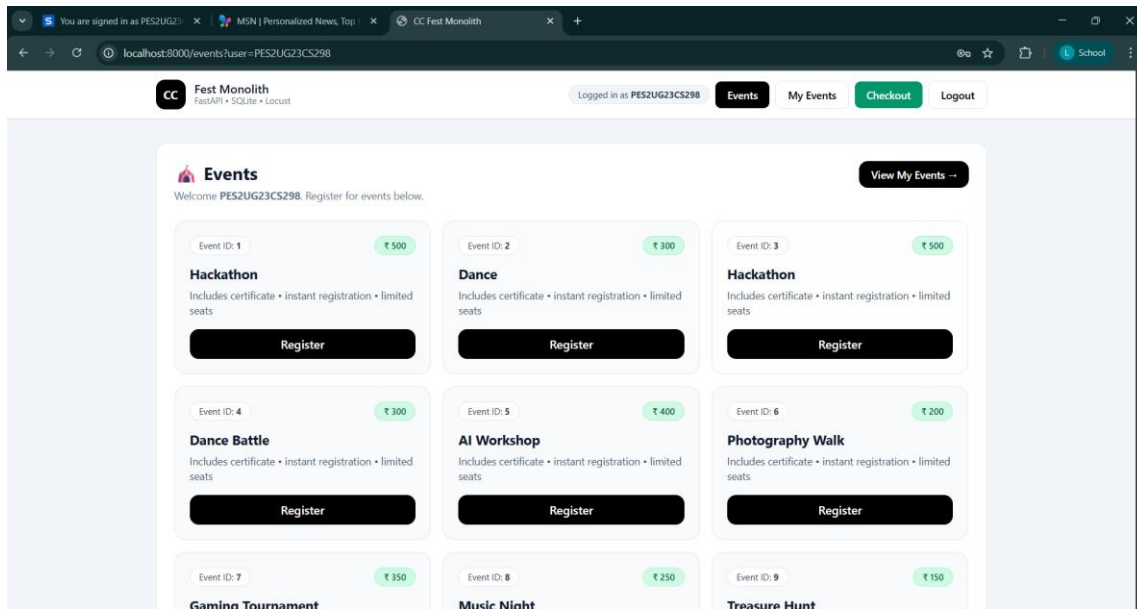
SRN: PES2UG23CS298

Github link

:https://github.com/PES2UG23CS298/Monolith_CC_Lab-2

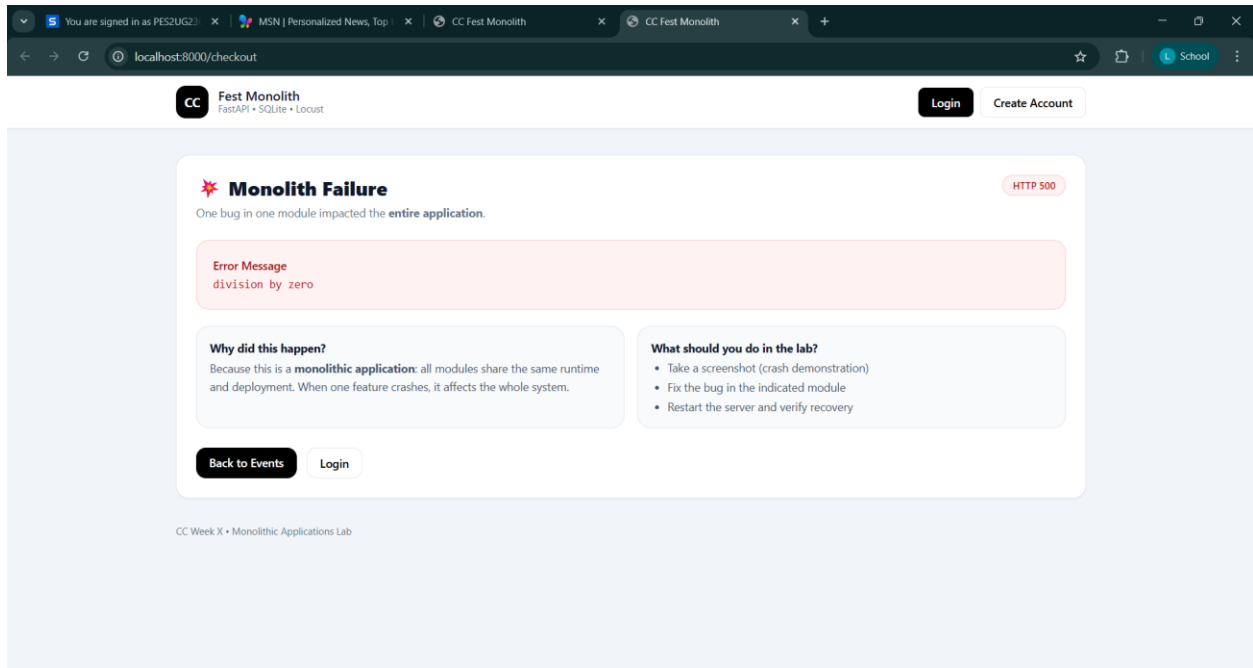
Screenshots:

Events

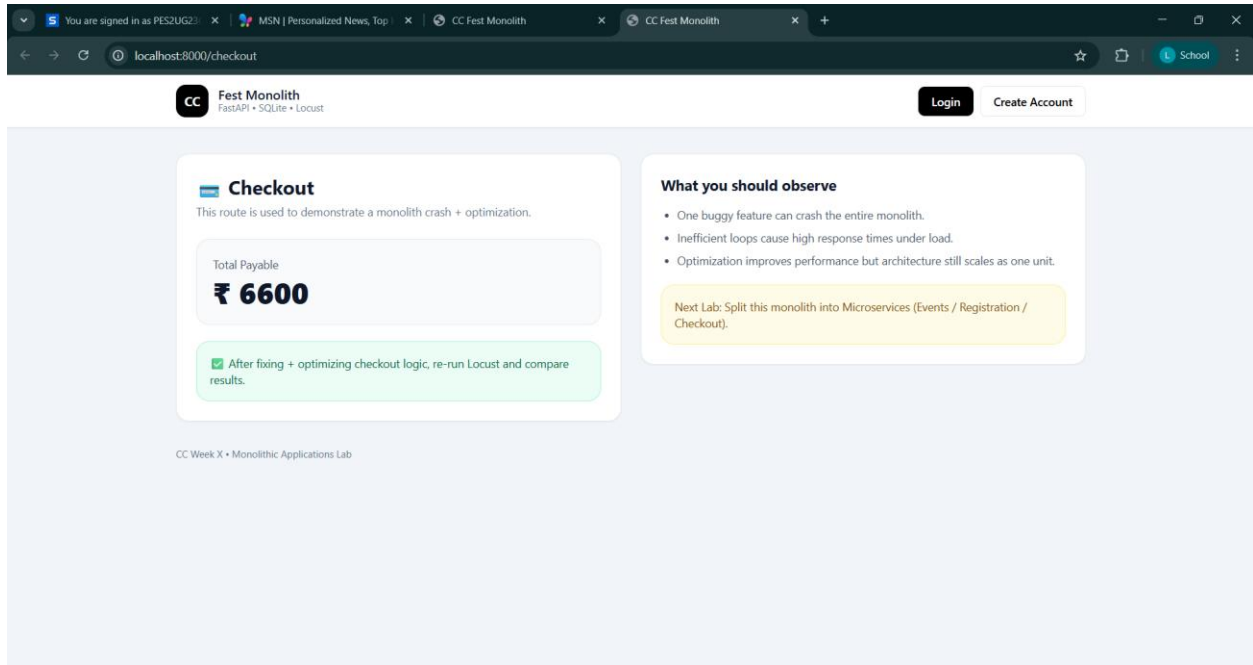


Monolith Crash

```
ZeroDivisionError: division by zero
INFO:      127.0.0.1:53725 - "GET /checkout HTTP/1.1" 500 Internal Server Error
ERROR:     Exception in ASGI application
```



Fix the Bug



```
INFO: 127.0.0.1:52829 - "GET /checkout HTTP/1.1" 200 OK
```

Load Testing using Locust

Before checkout Optimisation

LOCUST											Host	Status	RPS	Failures	NEW	RESET	
											http://localhost:8000	STOPPED	0.7	0%			
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	/checkout	19	0	8	2000	2000	113.99	3	2027	2797	0.7	0					
Aggregated		19	0	8	2000	2000	113.99	3	2027	2797	0.7	0					

```
File Edit Selection View Go Run Terminal Help
Q PES2UG23CS298

EXPLORER
PES2UG23CS298
  .venv
  CC Lab-2
  _pycache_
  checkout
  _pycache_
  _init_.py
  locust
  templates
  database.py
  fest.db
  insert_events.py
  main.py
  requirements.txt

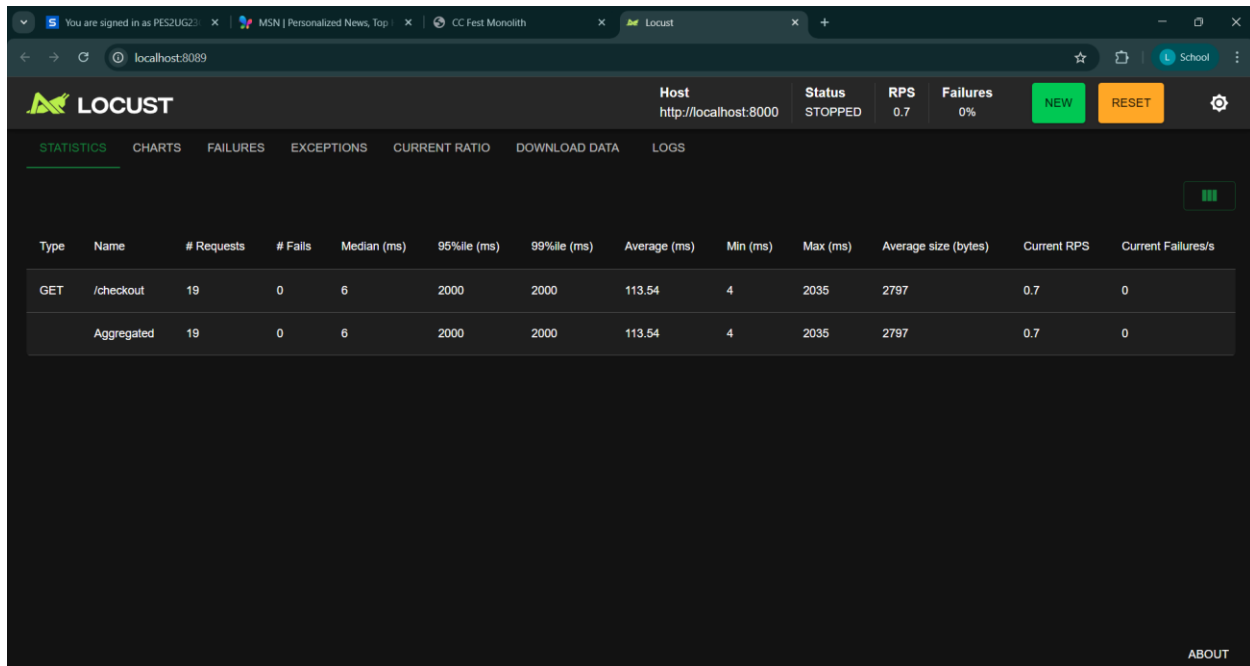
main.py
def checkout_logic():
    events = db.execute("SELECT fee FROM events").fetchall()
    # Uncomment this line initially for the crash screenshot task
    # 1 / 0

TERMINAL
(.venv) PS D:\LAB\PES2UG23CS298\CC Lab-2> locust -f locust/checkout_locustfile.py
[2026-01-30 15:19:22,006] LAPTOP-MK9GUSCH/INFO/locust.main: Starting web interface at http://localhost:8000, press enter to open your default browser.
[2026-01-30 15:19:33,692] LAPTOP-MK9GUSCH/INFO/locust.runners: Ramping to 1 users at a rate of 1.00 per second
[2026-01-30 15:19:33,693] LAPTOP-MK9GUSCH/INFO/locust.runners: All users spawned: {"CheckoutUser": 1} (1 total users)
Traceback (most recent call last):
  File "D:\LAB\PES2UG23CS298\.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
2026-01-30T09:51:45Z
[2026-01-30 15:21:45,820] LAPTOP-MK9GUSCH/INFO/locust.main: Shutting down (exit code 0)

Type      Name      # reqs      # fails      Avg      Min      Max      Med      req/s      failures/s
-----
GET /checkout 19          0(0.00%)    113       2      2026      8        0.66       0.00
Aggregated 19          0(0.00%)    113       2      2026      8        0.66       0.00

Response time percentiles (approximated)
Type      Name      % # reqs      50%      60%      75%      80%      90%      95%      98%      99.0%      99.9%      100
-----
GET /checkout 0 19          8        9        9        9        11      2000      2000      2000      2000      200
Aggregated 0 19          8        9        9        9        11      2000      2000      2000      2000      200
```

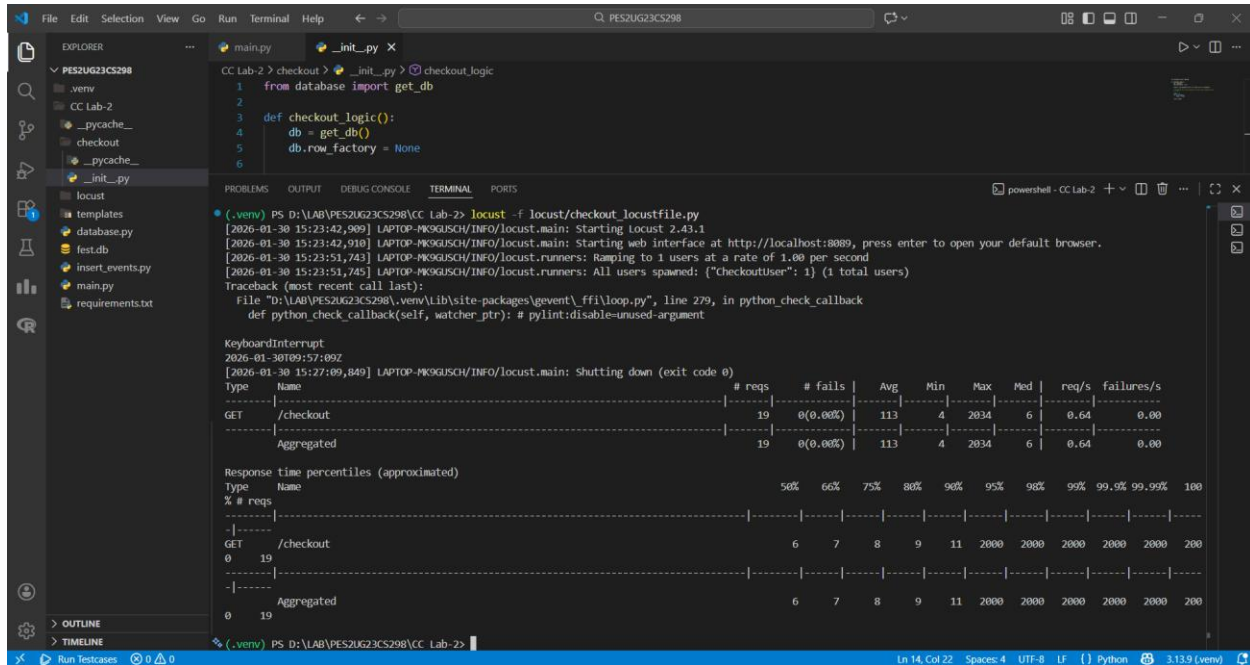
After checkout Optimisation



The screenshot shows the Locust web interface at `localhost:8089`. The interface includes a top navigation bar with tabs for STATISTICS, CHARTS, FAILURES, EXCEPTIONS, CURRENT RATIO, DOWNLOAD DATA, and LOGS. The main content area displays a table of statistics 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	19	0	6	2000	2000	113.54	4	2035	2797	0.7	0
Aggregated		19	0	6	2000	2000	113.54	4	2035	2797	0.7	0

At the bottom right, there is an 'ABOUT' link.



The screenshot shows a VS Code terminal window with the following output:

```
(.venv) PS D:\LAB\PES2UG23CS298\CC Lab-2> locust -f locust/checkout_locustfile.py
[2026-01-30 15:23:42,909] LAPTOP-MK9GUSCH/INFO/locust.main: Starting locust 2.43.1
[2026-01-30 15:23:42,910] LAPTOP-MK9GUSCH/INFO/locust.main: Starting web interface at http://localhost:8089, press enter to open your default browser.
[2026-01-30 15:23:51,743] LAPTOP-MK9GUSCH/INFO/locust.runners: Ramping to 1 users at a rate of 1.00 per second
[2026-01-30 15:23:51,745] LAPTOP-MK9GUSCH/INFO/locust.runners: All users spawned: ("Checkoutuser": 1) (1 total users)
Traceback (most recent call last):
  File "D:\LAB\PES2UG23CS298\.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
2026-01-30 15:27:09,849] LAPTOP-MK9GUSCH/INFO/locust.main: Shutting down (exit code 0)
```

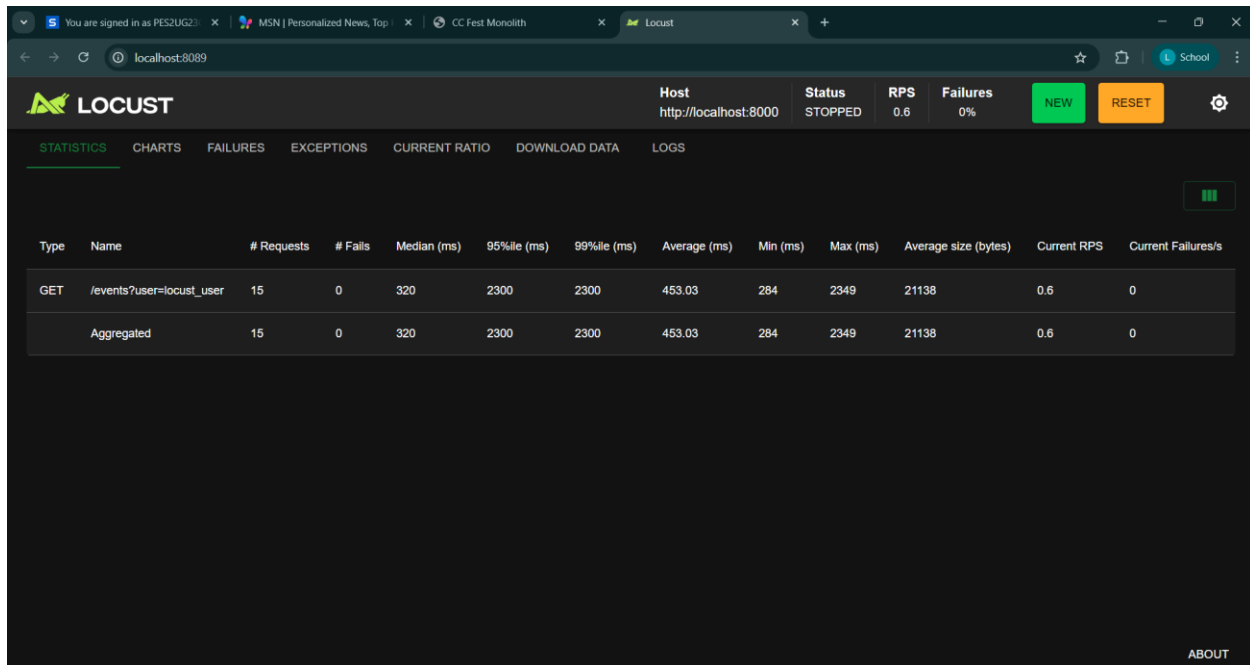
Below the terminal output, there is a table of statistics:

Type	Name	# reqs	# fails	Avg	Min	Max	Med	req/s	failures/s
GET	/checkout	19	0(0.00%)	113	4	2034	6	0.64	0.00
Aggregated		19	0(0.00%)	113	4	2034	6	0.64	0.00

Below the table, there is a section for response time percentiles (approximated):

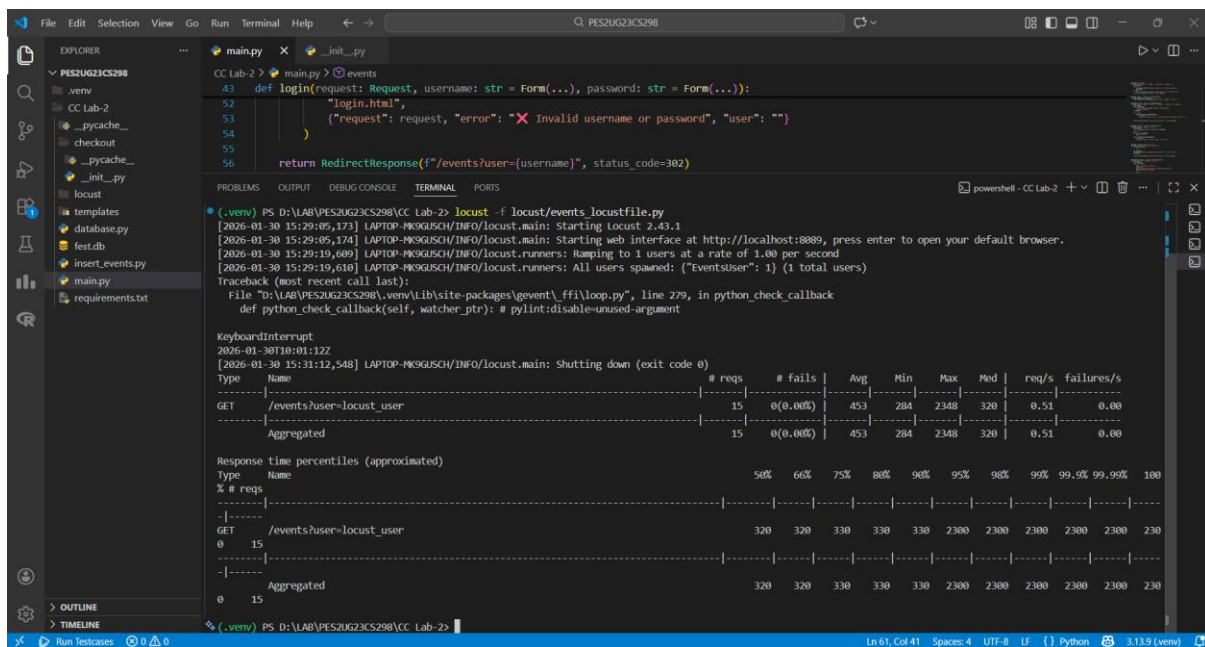
Type	Name	50%	60%	75%	80%	90%	95%	98%	99%	99.5%	99.9%	100%
GET	/checkout	6	7	8	9	11	2000	2000	2000	2000	2000	2000
Aggregated		6	7	8	9	11	2000	2000	2000	2000	2000	2000

Before events route optimisation



The screenshot shows the Locust web interface at localhost:8089. The status is STOPPED, RPS is 0.6, and Failures are 0%. The statistics table shows the following 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	15	0	320	2300	2300	453.03	284	2349	21138	0.6	0
	Aggregated	15	0	320	2300	2300	453.03	284	2349	21138	0.6	0



The screenshot shows a Python IDE with the Locust code and terminal output. The code defines a login function and a RedirectResponse. The terminal output shows the Locust interface starting at http://localhost:8089, ramping to 1 users at 1.00 per second, and displaying statistics for the /events?user=locust_user route.

```
def login(request: Request, username: str = Form(...), password: str = Form(...)):
    "login.html",
    {"request": request, "error": "Invalid username or password", "user": ""}
)
return RedirectResponse(f"/events?user={username}", status_code=302)
```

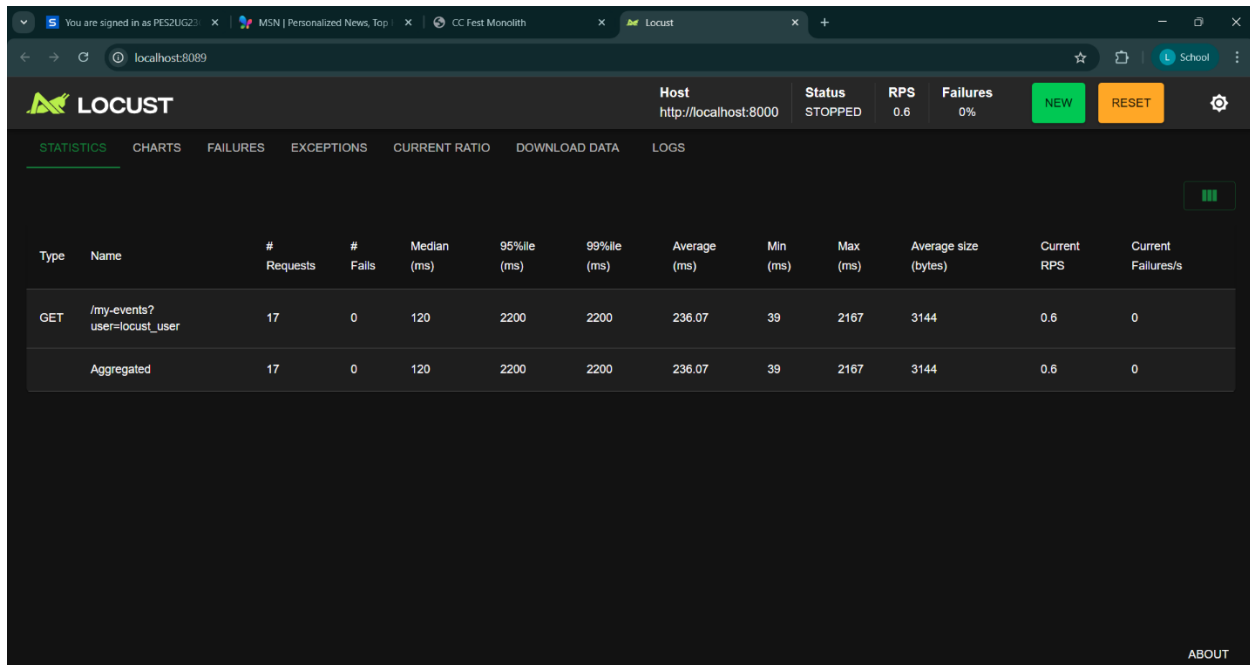
Terminal Output:

```
PS D:\LAB\PES2UG23CS298\CC Lab-2> locust -f locust/events_locustfile.py
[2026-01-30 15:29:05,173] LAPTOP-MKOGUSCH/INFO/locust.main: Starting locust 2.43.1
[2026-01-30 15:29:05,174] LAPTOP-MKOGUSCH/INFO/locust.main: Starting web interface at http://localhost:8089, press enter to open your default browser.
[2026-01-30 15:29:19,609] LAPTOP-MKOGUSCH/INFO/locust.runners: Ramping to 1 users at a rate of 1.00 per second
[2026-01-30 15:29:19,610] LAPTOP-MKOGUSCH/INFO/locust.runners: All users spawned: ("EventsUser": 1) (1 total users)
Traceback (most recent call last):
  File "D:\LAB\PES2UG23CS298\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
2026-01-30T10:01:12Z
[2026-01-30 15:31:12,548] LAPTOP-MKOGUSCH/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                15         0(0.00%)  453    284    2348    320    0.51     0.00
-----
Aggregated                                     15         0(0.00%)  453    284    2348    320    0.51     0.00
-----

Response time percentiles (approximated)
Type      Name                                     50%    66%    75%    80%    90%    95%    98%    99.5%  99.95%  100
-----
GET       /events?user=locust_user                320    320    330    330    330    2300    2300    2300    2300    230
-----
Aggregated                                     320    320    330    330    330    2300    2300    2300    2300    230
-----
0       15                                     320    320    330    330    330    2300    2300    2300    2300    230
```

After events route optimisation



The screenshot shows a VS Code editor with the following files in the Explorer:

- .venv
- CC Lab-2
- _pycache_
- checkout
- _pycache_
- _init_.py
- locust
- templates
- database.py
- test.db
- insert_events.py
- main.py
- requirements.txt

The main.py file is open, showing the following code:

```
43 def login(request, username: str = Form(...), password: str = Form(...)):
44     """login.html"""
45     ("request": request, "error": "Invalid username or password, "user": ""})
46 )
47 )
48 )
49 return RedirectResponse(f"/events?user={username}", status_code=302)
```

The terminal output shows the following commands and results:

```
PS D:\LAB\PES2UG23CS298\CC Lab-2> locust -f locust/events_locustfile.py
[2026-01-30 15:33:16,424] LAPTOP-MK9GUSCH/INFO/locust.main: Starting locust 2.43.1
[2026-01-30 15:33:16,424] LAPTOP-MK9GUSCH/INFO/locust.main: Starting web interface at http://localhost:8089, press enter to open your default browser.
[2026-01-30 15:33:25,397] LAPTOP-MK9GUSCH/INFO/locust.runners: Ramping to 1 users at a rate of 1.00 per second
[2026-01-30 15:33:25,400] LAPTOP-MK9GUSCH/INFO/locust.runners: All users spawned: {"Eventsuser": 1} (1 total users)
Traceback (most recent call last):
  File "D:\LAB\PES2UG23CS298\.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
2026-01-30 15:34:04,242
[2026-01-30 15:34:24,727] LAPTOP-MK9GUSCH/INFO/locust.main: Shutting down (exit code 0)
```

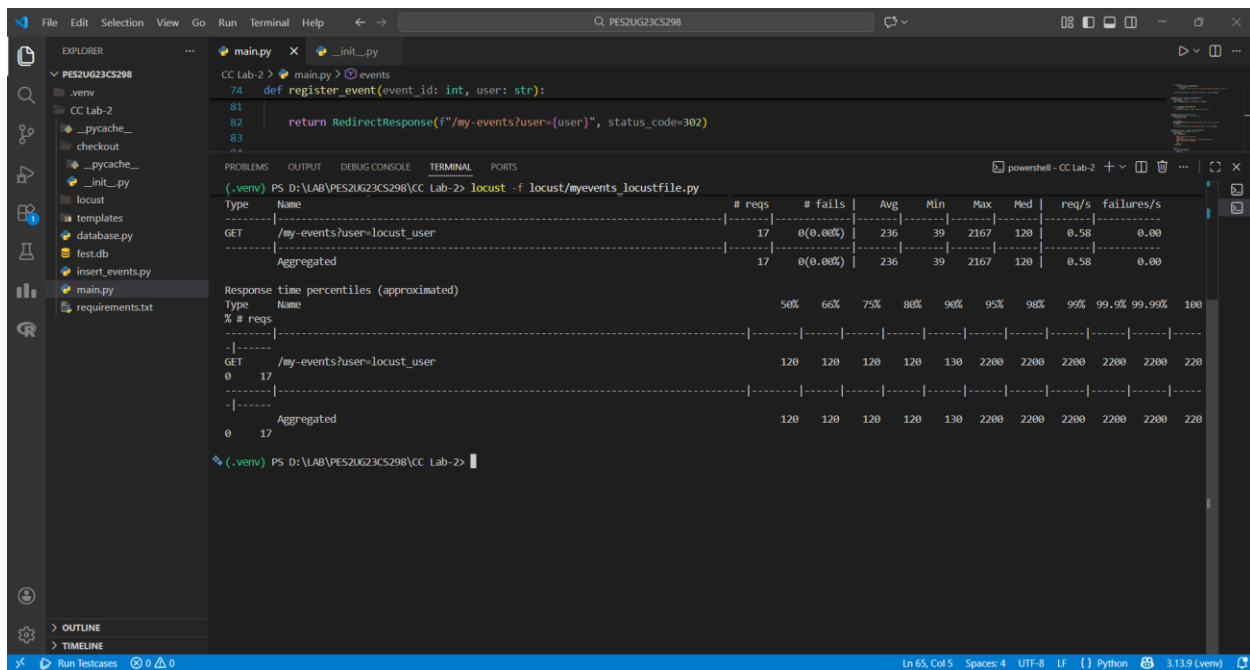
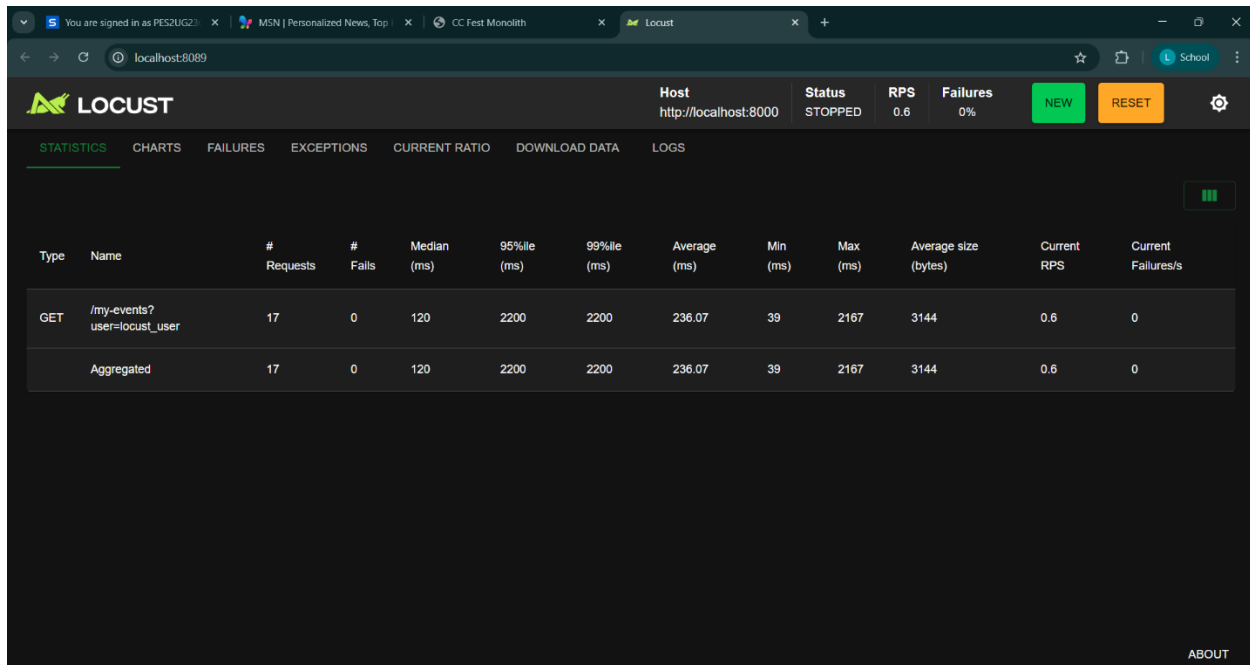
The terminal also displays a summary of the test results:

Type	Name	# reqs	# fails	Avg	Min	Max	Med	req/s	failures/s
GET	/events?user=locust_user	19	0(0.00%)	113	3	2018	7	0.65	0.00
Aggregated		19	0(0.00%)	113	3	2018	7	0.65	0.00

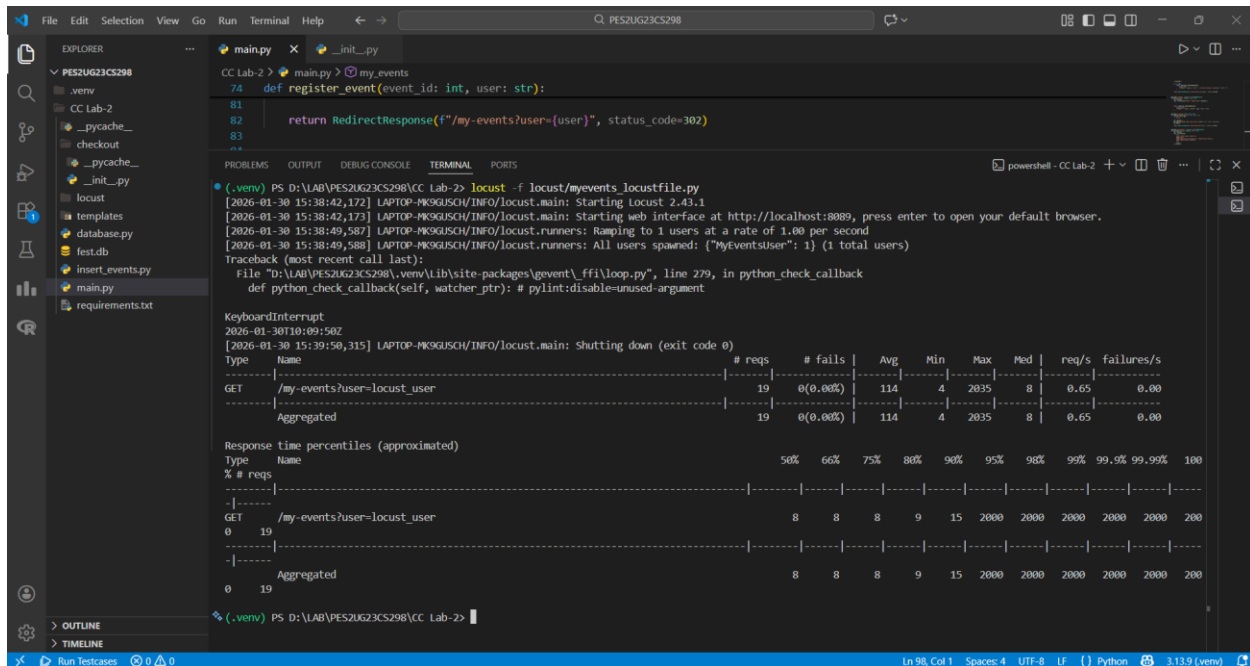
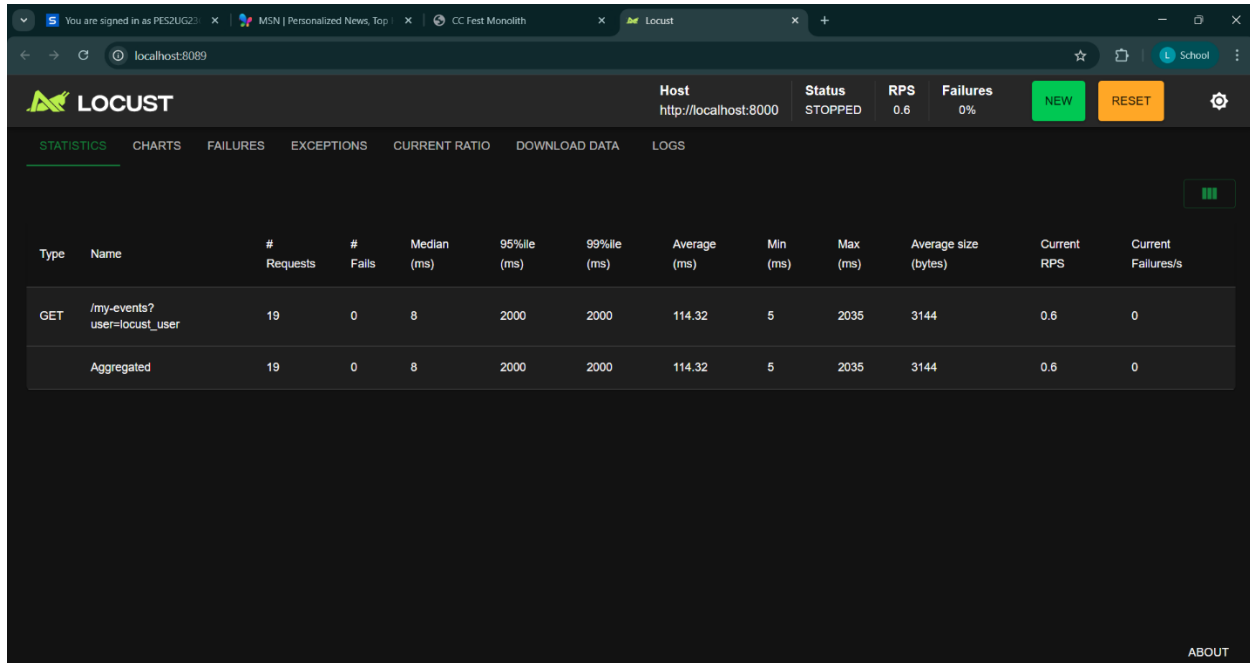
The terminal also shows response time percentiles (approximated) for the GET /events?user=locust_user route:

Type	Name	50%	66%	75%	80%	90%	95%	98%	99%	99.9%	99.99%	100%
GET	/events?user=locust_user	7	8	9	9	11	2000	2000	2000	2000	2000	200
Aggregated		7	8	9	9	11	2000	2000	2000	2000	2000	200

Before my-events route optimisation



After my-events route optimisation



Checkout Load Test (Before Optimization)

Observation:

- Average response time is relatively high
- Requests per second (RPS) are low
- Performance degrades due to inefficient computation

Reason:

The checkout logic uses an inefficient loop-based calculation.

Checkout Load Test (After Optimization)

What changed?

The inefficient loop was replaced with a direct aggregation over event prices.

Result:

- Average response time decreased
- RPS remained approximately the same

Conclusion:

The route became faster due to reduced computational overhead, **even though system capacity stayed constant.**

Route: /events

Bottleneck Identified:

The /events route contained a dummy loop that performed unnecessary iterations without contributing to the response. This caused extra CPU usage and increased response time under load.

Optimization Performed:

The dummy loop was **removed completely**, allowing the route to return event data directly without redundant processing.

Why Performance Improved:

By eliminating unnecessary computation, the CPU workload per request was reduced. This resulted in **lower response time** and improved performance during load testing.

Route: /my-events

Bottleneck Identified:

Similar to the /events route, /my-events contained a dummy loop that executed pointlessly for every request, increasing execution time.

Optimization Performed:

The dummy loop was **removed**, and only the required logic for fetching user-specific events was retained.

Why Performance Improved:

Removing the redundant loop reduced processing overhead, making request handling faster and improving response times under concurrent load.

Final Conclusion

- In both routes, performance improved because unnecessary dummy loops were removed, reducing computational overhead and making the monolithic application more efficient under load.
- Monolithic applications are easy to build and deploy
- They suffer from poor fault isolation
- Performance optimization helps, but scalability and reliability remain limited
- Microservices solve these issues by isolating failures and scaling independently