CC LAB 2

NAME : CHARAN L

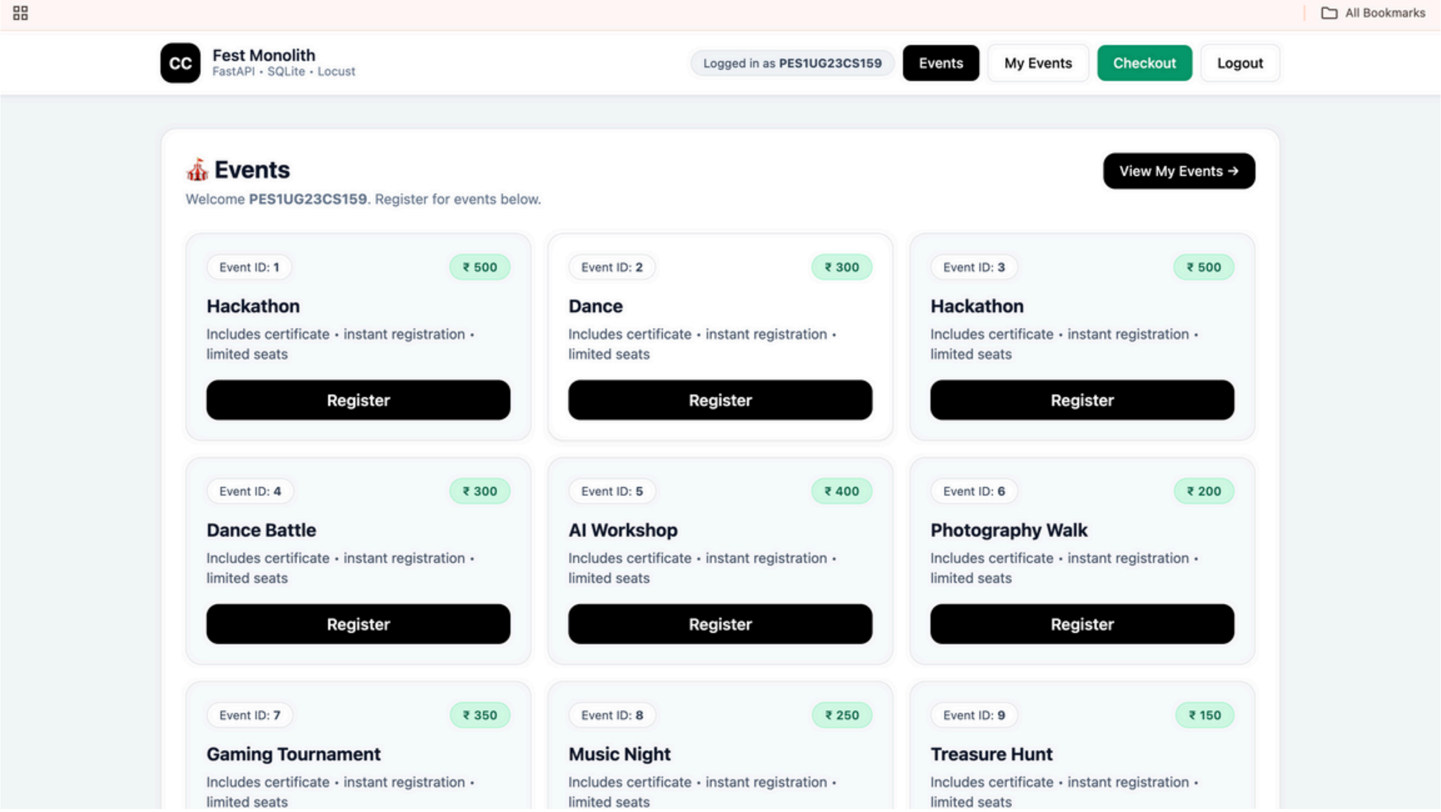
SRN : PES1UG23CS159

SECTION : C

GITHUB LINK : GitHub - Charan-1810/PES1UG23CS159_CCLAB-2

SCREENSHOTS:

SS1



Events

Welcome **PES1UG23CS159**. Register for events below.

Event ID: 1 — ₹ 500
Hackathon
Includes certificate • instant registration • limited seats
Register

Event ID: 2 — ₹ 300
Dance
Includes certificate • instant registration • limited seats
Register

Event ID: 3 — ₹ 500
Hackathon
Includes certificate • instant registration • limited seats
Register

Event ID: 4 — ₹ 300
Dance Battle
Includes certificate • instant registration • limited seats
Register

Event ID: 5 — ₹ 400
AI Workshop
Includes certificate • instant registration • limited seats
Register

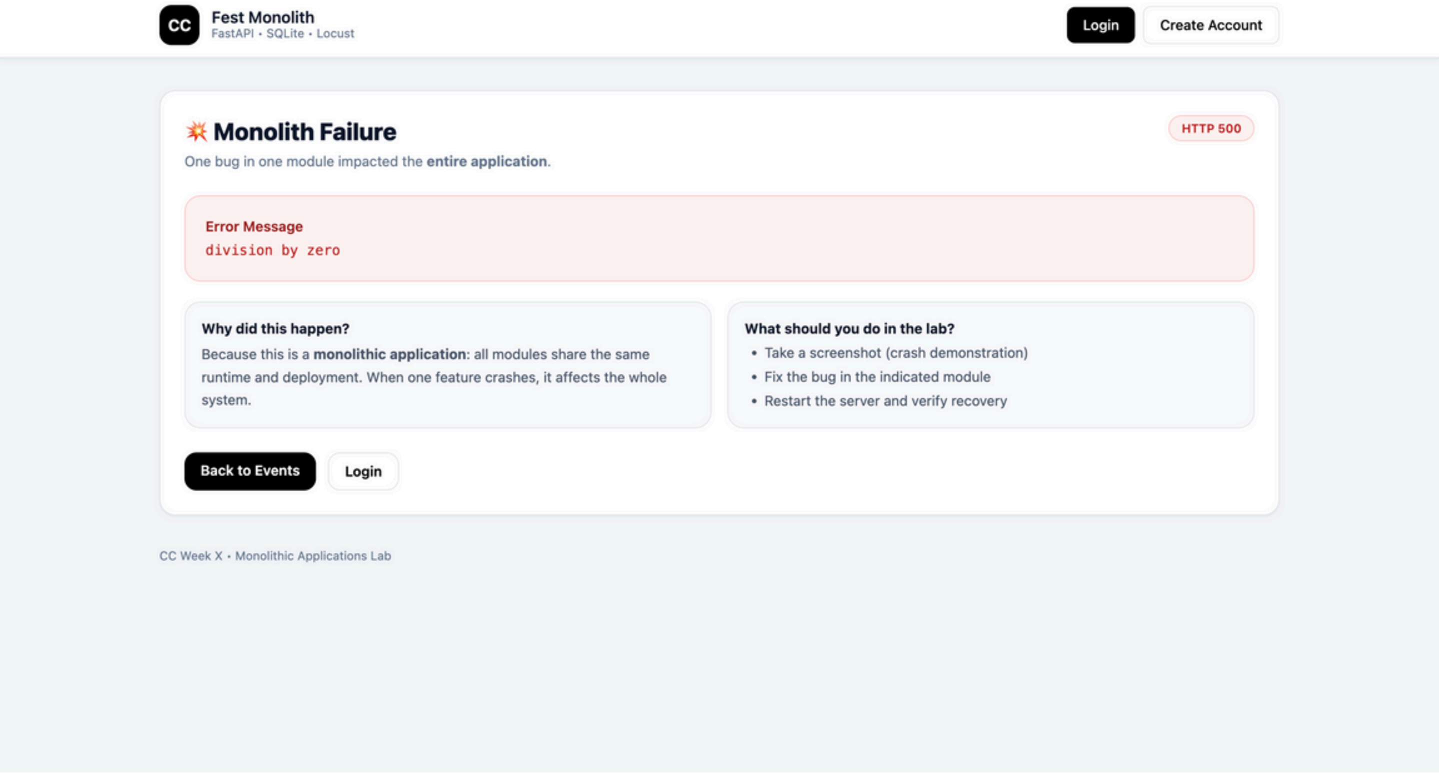
Event ID: 6 — ₹ 200
Photography Walk
Includes certificate • instant registration • limited seats
Register

Event ID: 7 — ₹ 350
Gaming Tournament
Includes certificate • instant registration • limited seats

Event ID: 8 — ₹ 250
Music Night
Includes certificate • instant registration • limited seats

Event ID: 9 — ₹ 150
Treasure Hunt
Includes certificate • instant registration • limited seats

SS2:



💥 Monolith Failure

HTTP 500

One bug in one module impacted the **entire application**.

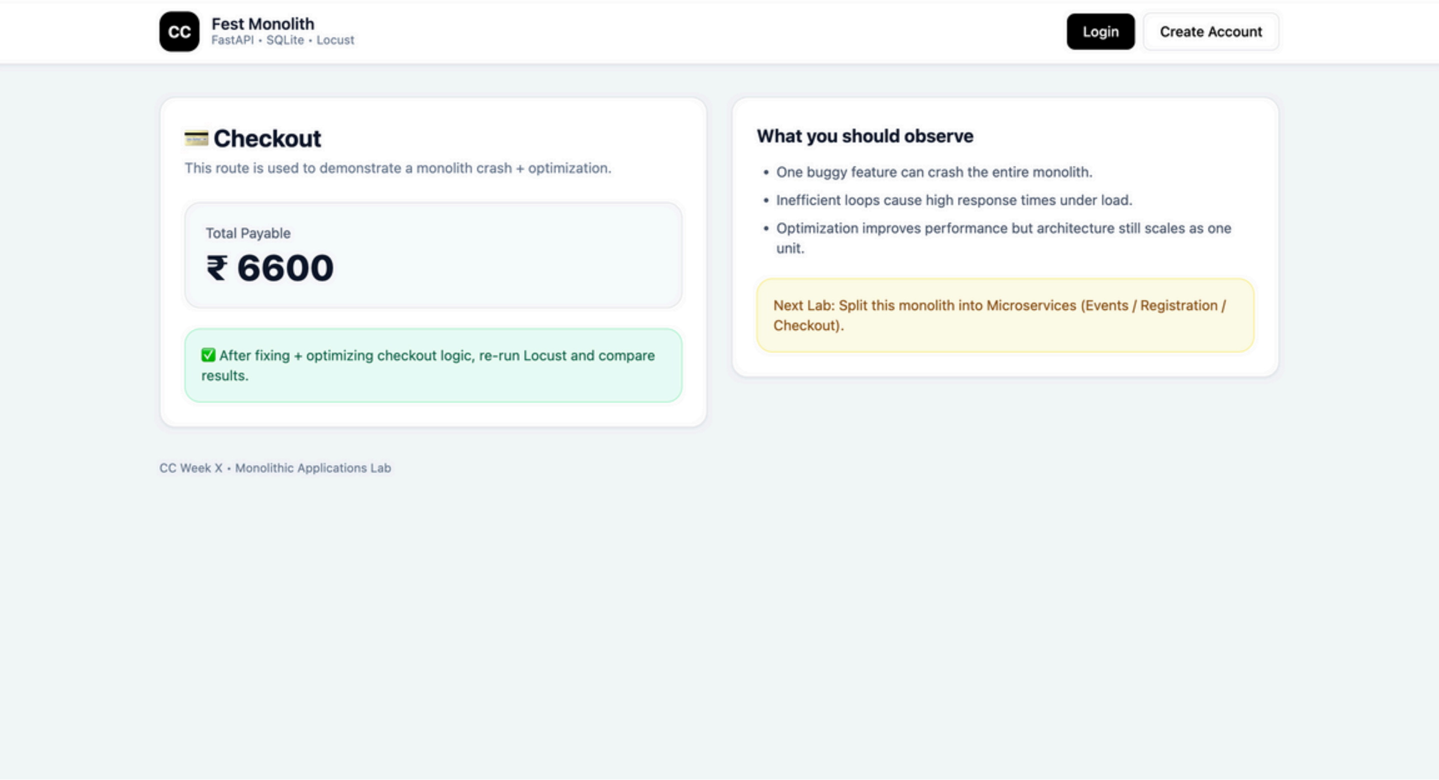
Error Message
`division by zero`

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

CC Week X • Monolithic Applications Lab

SS3:



💳 Checkout

This route is used to demonstrate a monolith crash + optimization.

Total Payable
₹ 6600

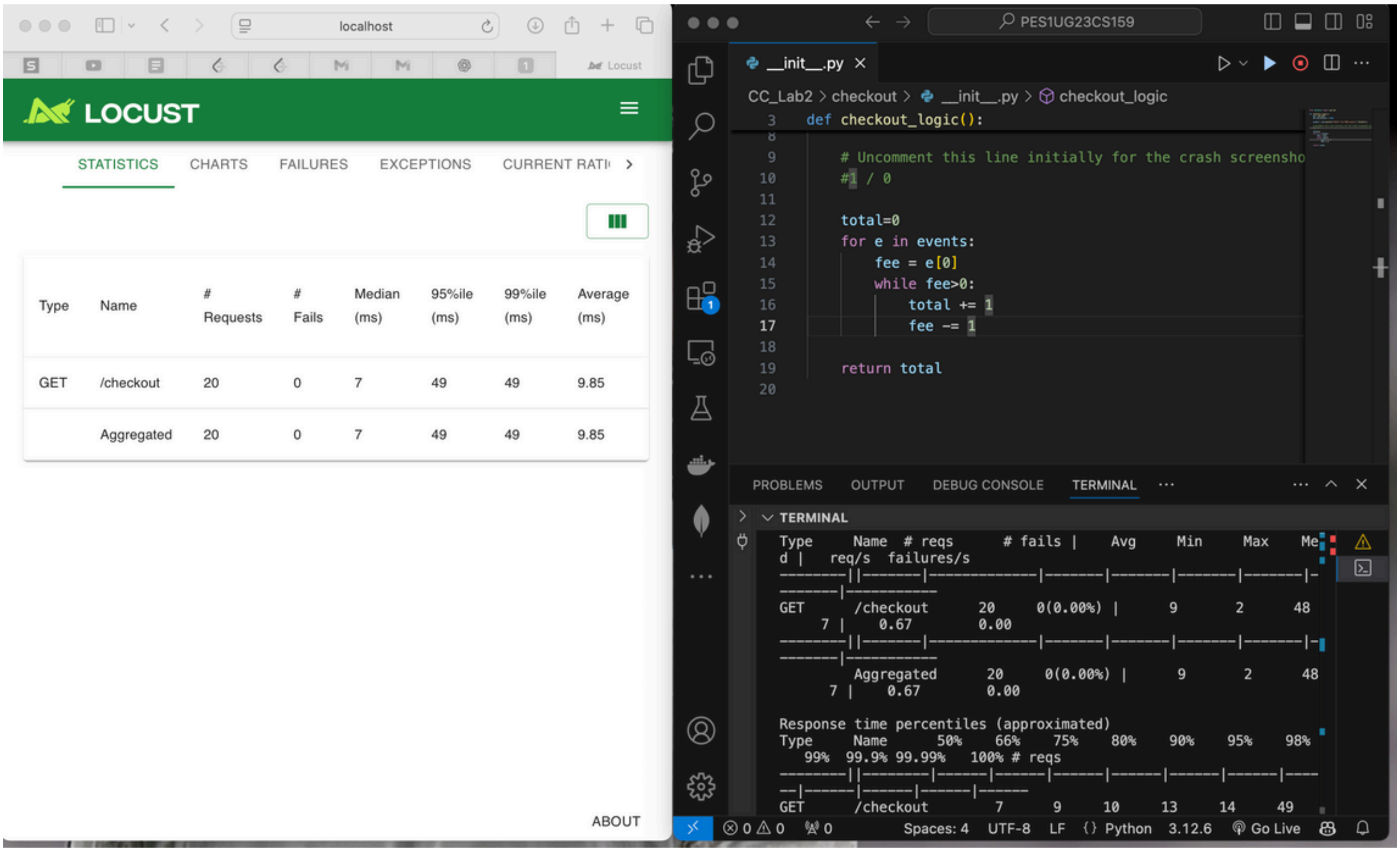
✅ After fixing + optimizing checkout logic, re-run Locust and compare results.

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

SS4:

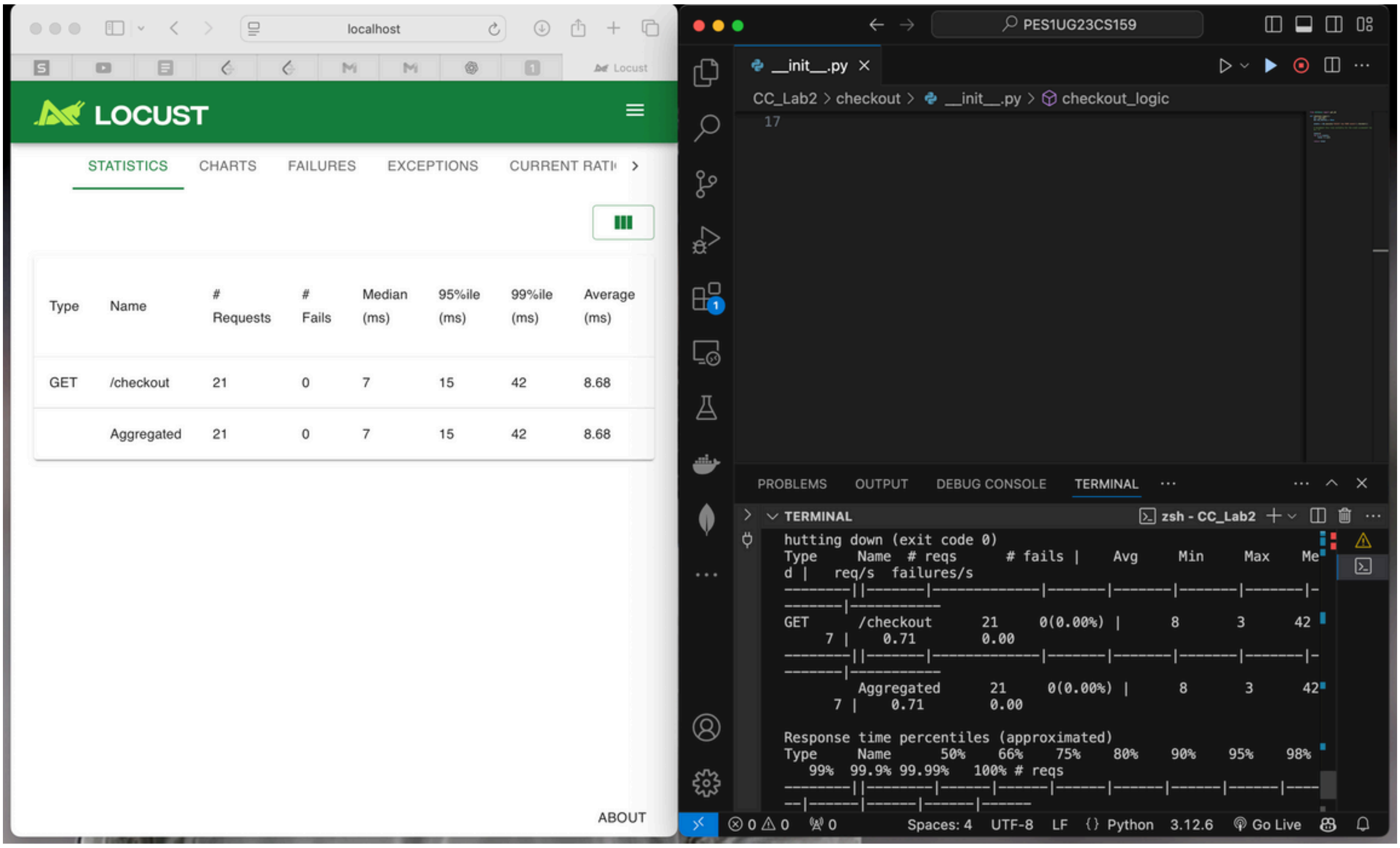


Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)
GET	/checkout	20	0	7	49	49	9.85
	Aggregated	20	0	7	49	49	9.85

```python
def checkout_logic():

    # Uncomment this line initially for the crash screensho
    #1 / 0

    total=0
    for e in events:
        fee = e[0]
        while fee>0:
            total += 1
            fee -= 1

    return total
```

```
Type     Name  # reqs      # fails |    Avg     Min     Max    Me
d |   req/s  failures/s
--------||-------|-------------|-------|-------|-------|-------|-
-------|-----------
GET      /checkout      20     0(0.00%) |      9       2      48
     7 |    0.67       0.00
--------||-------|-------------|-------|-------|-------|-------|-
-------|-----------
         Aggregated      20     0(0.00%) |      9       2      48
      7 |    0.67       0.00

Response time percentiles (approximated)
Type     Name       50%    66%    75%    80%    90%    95%    98%
   99%  99.9% 99.99%   100% # reqs
--------||--------|------|------|------|------|------|------|----
--|------|------|------|------
GET      /checkout       7      9     10     13     14     49
```

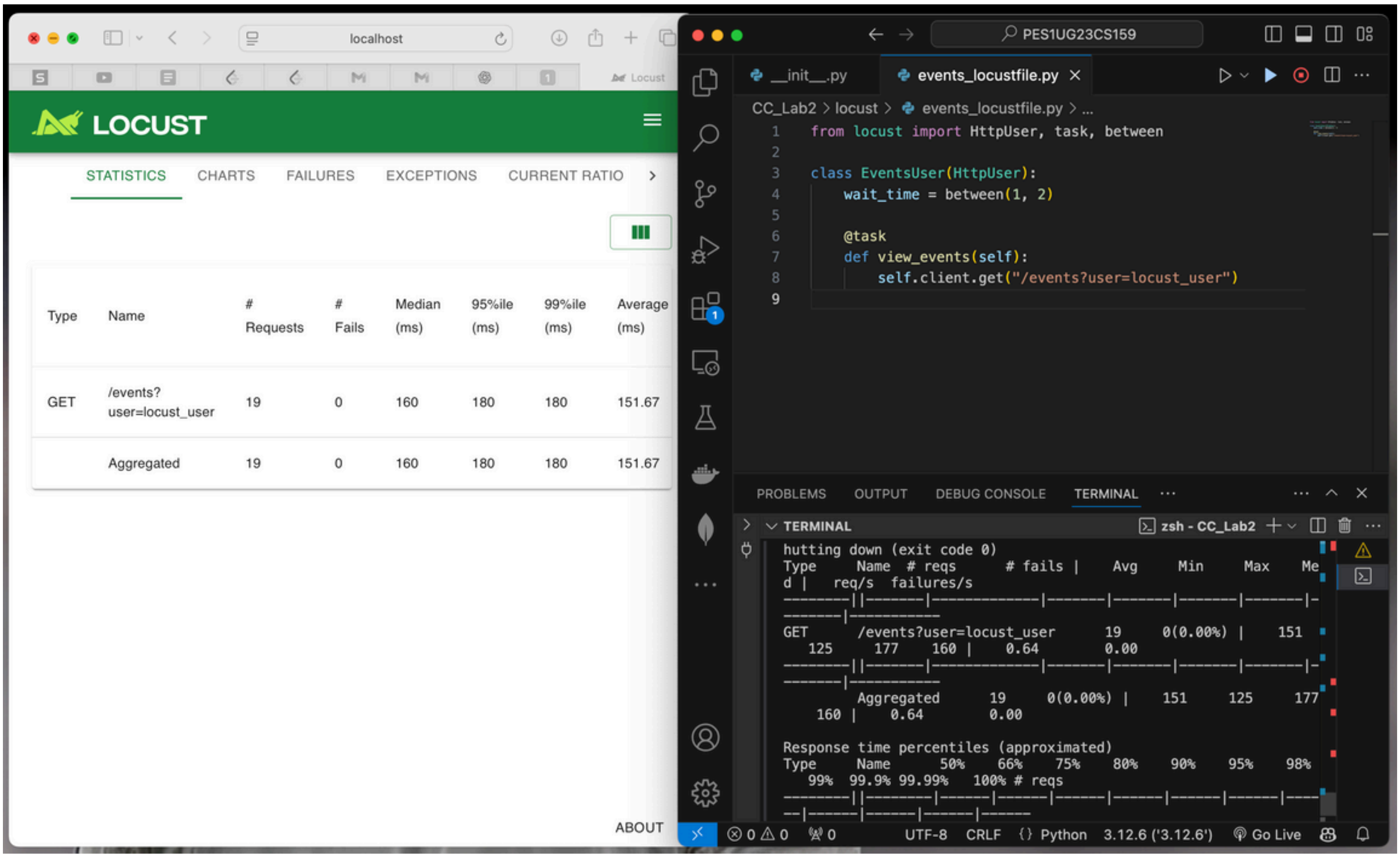
SS5:



Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)
GET	/checkout	21	0	7	15	42	8.68
	Aggregated	21	0	7	15	42	8.68

```
hutting down (exit code 0)
Type     Name  # reqs      # fails |    Avg     Min     Max    Me
d |   req/s  failures/s
--------||-------|-------------|-------|-------|-------|-------|-
-------|-----------
GET      /checkout      21     0(0.00%) |      8       3      42
     7 |    0.71       0.00
--------||-------|-------------|-------|-------|-------|-------|-
-------|-----------
         Aggregated      21     0(0.00%) |      8       3      42
      7 |    0.71       0.00

Response time percentiles (approximated)
Type     Name       50%    66%    75%    80%    90%    95%    98%
   99%  99.9% 99.99%   100% # reqs
--------||--------|------|------|------|------|------|------|----
--|------|------|------|------
```

SS6:

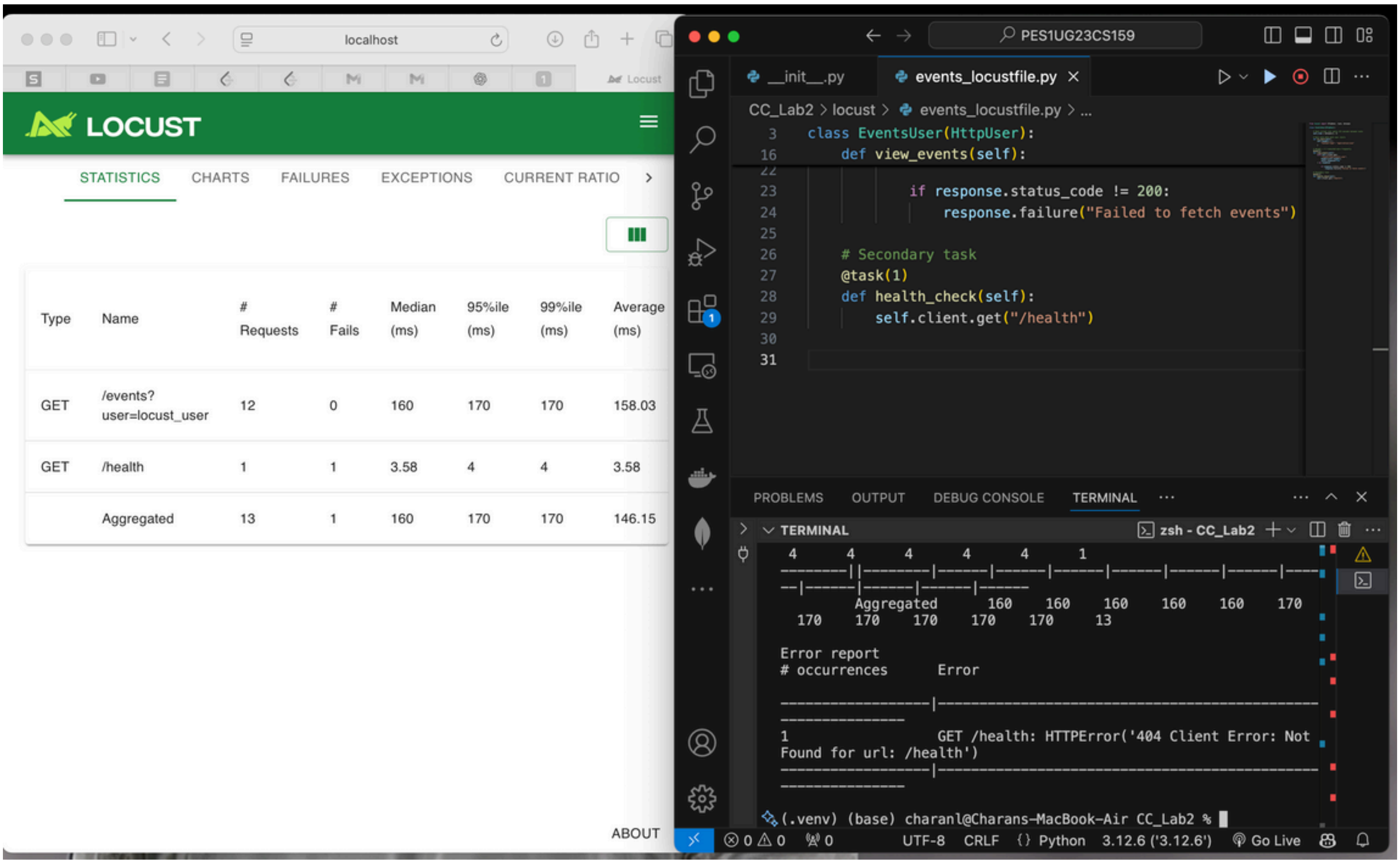


Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)
GET	/events? user=locust_user	19	0	160	180	180	151.67
	Aggregated	19	0	160	180	180	151.67

```python
from locust import HttpUser, task, between

class EventsUser(HttpUser):
    wait_time = between(1, 2)

    @task
    def view_events(self):
        self.client.get("/events?user=locust_user")

```

```
hutting down (exit code 0)
Type     Name  # reqs      # fails |    Avg     Min     Max    Me
d |   req/s  failures/s
--------||-------|-------------|-------|-------|-------|-------|-
-------|-----------
GET      /events?user=locust_user      19     0(0.00%) |    151
   125     177    160 |    0.64        0.00
--------||-------|-------------|-------|-------|-------|-------|-
-------|-----------
         Aggregated      19     0(0.00%) |    151     125     177
    160 |    0.64        0.00

Response time percentiles (approximated)
Type     Name        50%    66%    75%    80%    90%    95%    98%
   99%  99.9% 99.99%   100% # reqs
--------||---------|-------|-------|-------|-------|-------|-------|----
--|-------|-------|-------|-------
```

SS7:

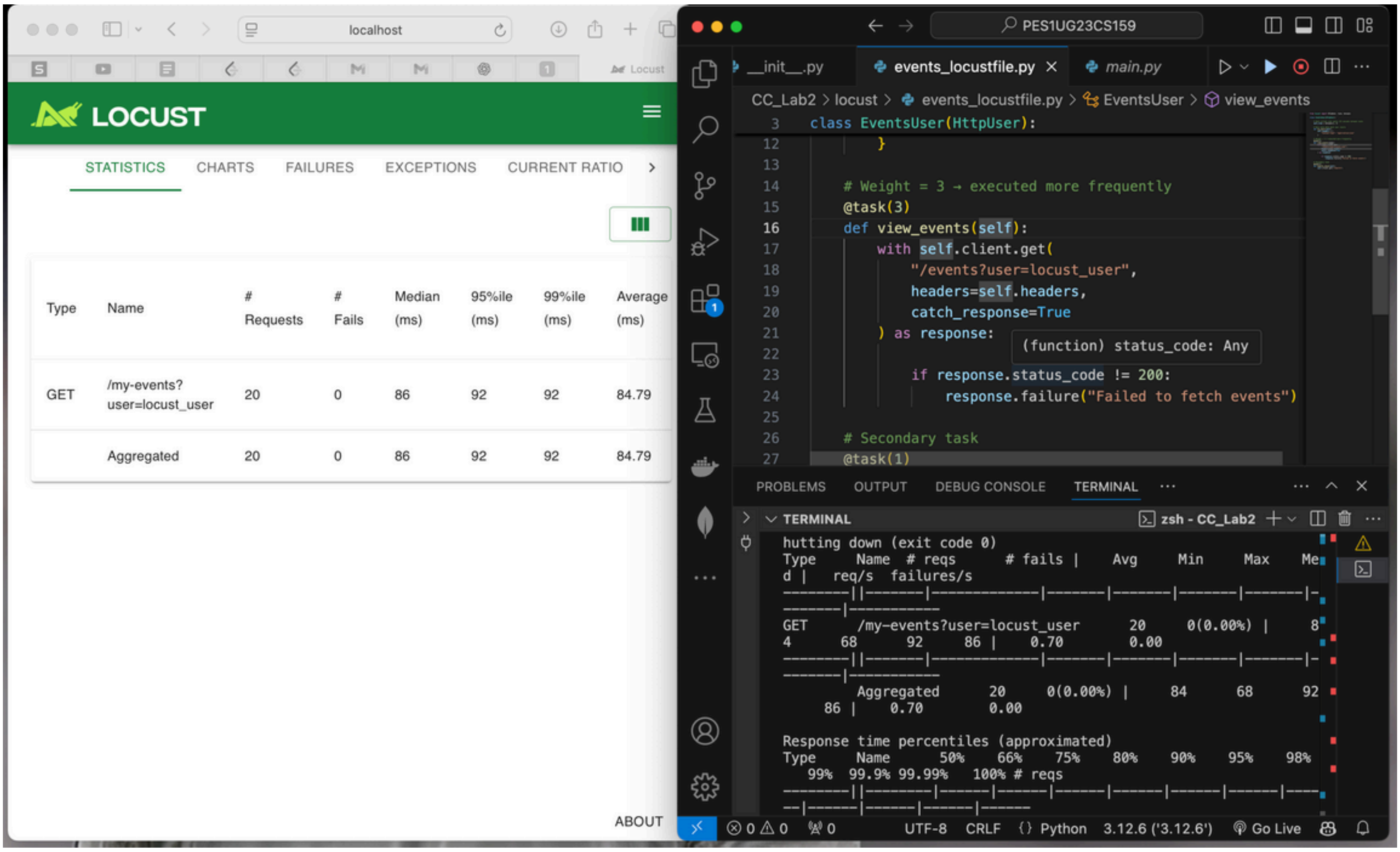


Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)
GET	/events? user=locust_user	12	0	160	170	170	158.03
GET	/health	1	1	3.58	4	4	3.58
	Aggregated	13	1	160	170	170	146.15

```python
class EventsUser(HttpUser):
    def view_events(self):
            if response.status_code != 200:
                response.failure("Failed to fetch events")

    # Secondary task
    @task(1)
    def health_check(self):
        self.client.get("/health")

```

```
 4       4       4       4       4       1
--------||---------|-------|-------|-------|-------|-------|-------|----
--|-------|-------|-------|-------
         Aggregated      160     160     160     160     160     170
   170     170     170     170     170      13

Error report
# occurrences      Error

------------------|----------------------------------------------------
---------------
1                  GET /health: HTTPError('404 Client Error: Not
Found for url: /health')
------------------|----------------------------------------------------
---------------

(.venv) (base) charanl@Charans-MacBook-Air CC_Lab2 %
```

SS8:

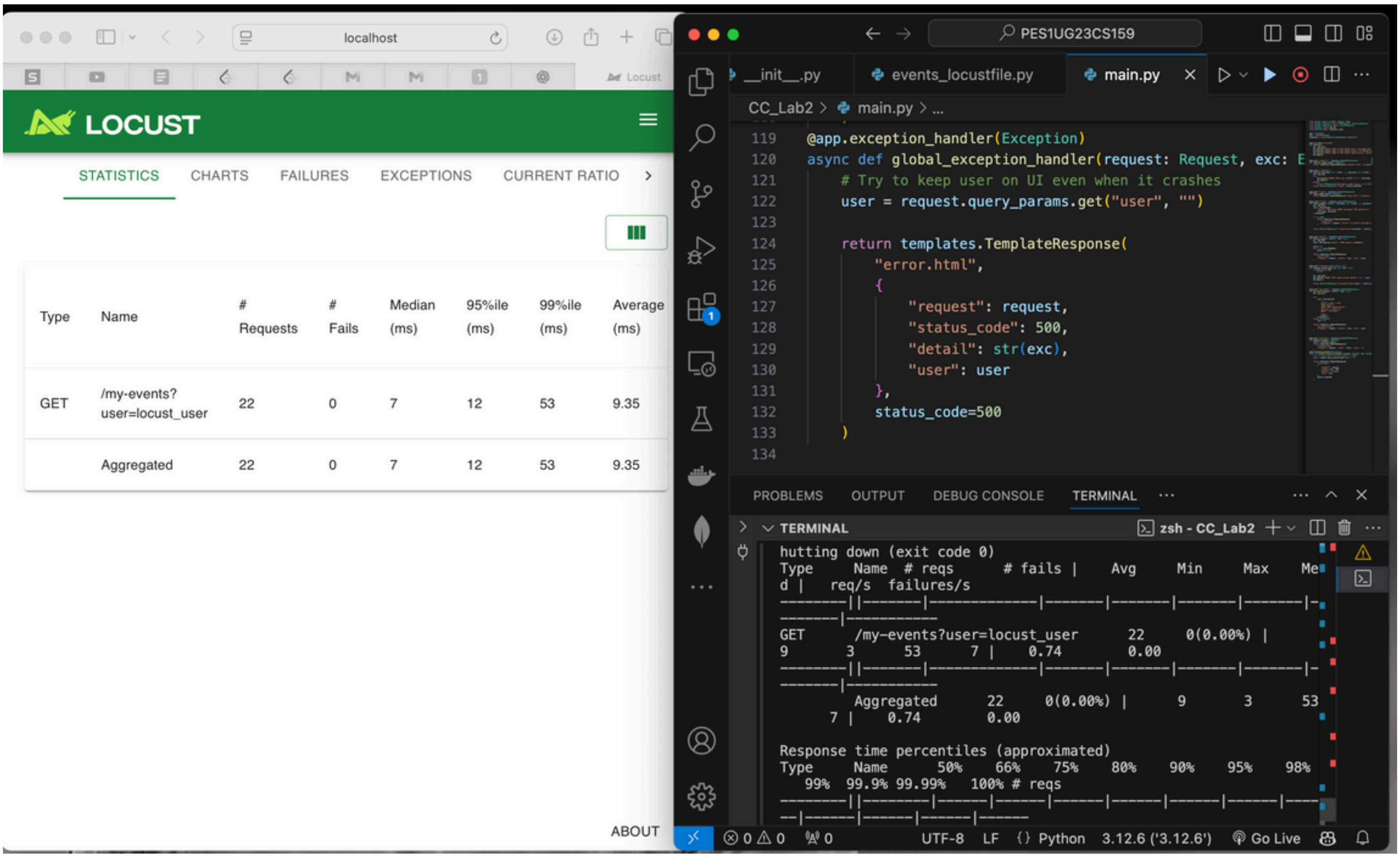


Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)
GET	/my-events?user=locust_user	20	0	86	92	92	84.79
	Aggregated	20	0	86	92	92	84.79

```python
class EventsUser(HttpUser):
        }

    # Weight = 3 → executed more frequently
    @task(3)
    def view_events(self):
        with self.client.get(
            "/events?user=locust_user",
            headers=self.headers,
            catch_response=True
        ) as response:
            if response.status_code != 200:
                response.failure("Failed to fetch events")

    # Secondary task
    @task(1)
```

```
hutting down (exit code 0)
Type     Name  # reqs      # fails |    Avg     Min     Max    Me
d |   req/s  failures/s
--------||-------|-------------|-------|-------|-------|-------|-
-------|-----------
GET      /my-events?user=locust_user      20     0(0.00%) |      8
4      68      92      86 |    0.70       0.00
--------||-------|-------------|-------|-------|-------|-------|-
-------|-----------
         Aggregated      20     0(0.00%) |      84      68      92
      86 |    0.70       0.00

Response time percentiles (approximated)
Type     Name        50%    66%    75%    80%    90%    95%    98%
   99%  99.9% 99.99%   100% # reqs
--------||--------|------|------|------|------|------|------|----
--|------|------|------|------
```

SS9:



Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)
GET	/my-events?user=locust_user	22	0	7	12	53	9.35
	Aggregated	22	0	7	12	53	9.35

```python
@app.exception_handler(Exception)
async def global_exception_handler(request: Request, exc: E
    # Try to keep user on UI even when it crashes
    user = request.query_params.get("user", "")

    return templates.TemplateResponse(
        "error.html",
        {
            "request": request,
            "status_code": 500,
            "detail": str(exc),
            "user": user
        },
        status_code=500
    )
```

```
hutting down (exit code 0)
Type     Name  # reqs      # fails |    Avg     Min     Max    Me
d |   req/s  failures/s
--------||-------|-------------|-------|-------|-------|-------|-
-------|-----------
GET      /my-events?user=locust_user      22     0(0.00%) |
9      3      53      7 |    0.74       0.00
--------||-------|-------------|-------|-------|-------|-------|-
-------|-----------
         Aggregated      22     0(0.00%) |      9       3      53
      7 |    0.74       0.00

Response time percentiles (approximated)
Type     Name        50%    66%    75%    80%    90%    95%    98%
   99%  99.9% 99.99%   100% # reqs
--------||--------|------|------|------|------|------|------|----
--|------|------|------|------
```

QUESTIONS:

1.What was the bottleneck?

The bottleneck was **inefficient data processing and repeated looping over event records**, which caused unnecessary CPU usage and increased response time during load testing.

2.What change did you make?

I optimized the logic by:

- Removing unnecessary loops
- Using direct aggregation instead of incremental counting
- Reducing redundant operations while fetching and processing event data

3.Why did the performance improve?

Performance improved because:

- Fewer iterations were executed
- CPU workload was reduced
- The response was generated using optimized logic

This decreased average response time and improved overall throughput

4.Explanation about optimization

Optimization was done by removing inefficient loops and redundant computations, and by simplifying data processing logic. These changes reduced CPU usage and execution time per request. As a result, average response time decreased while handling the same number of requests, demonstrating improved performance of the monolithic application.