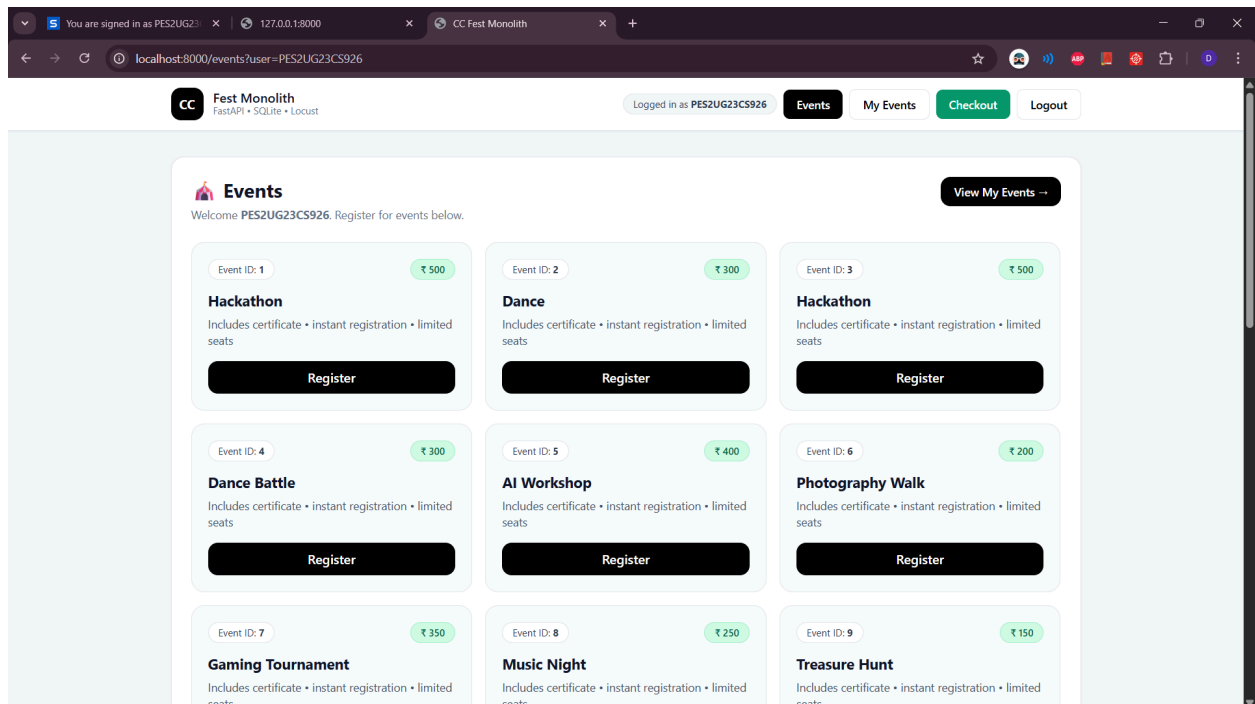


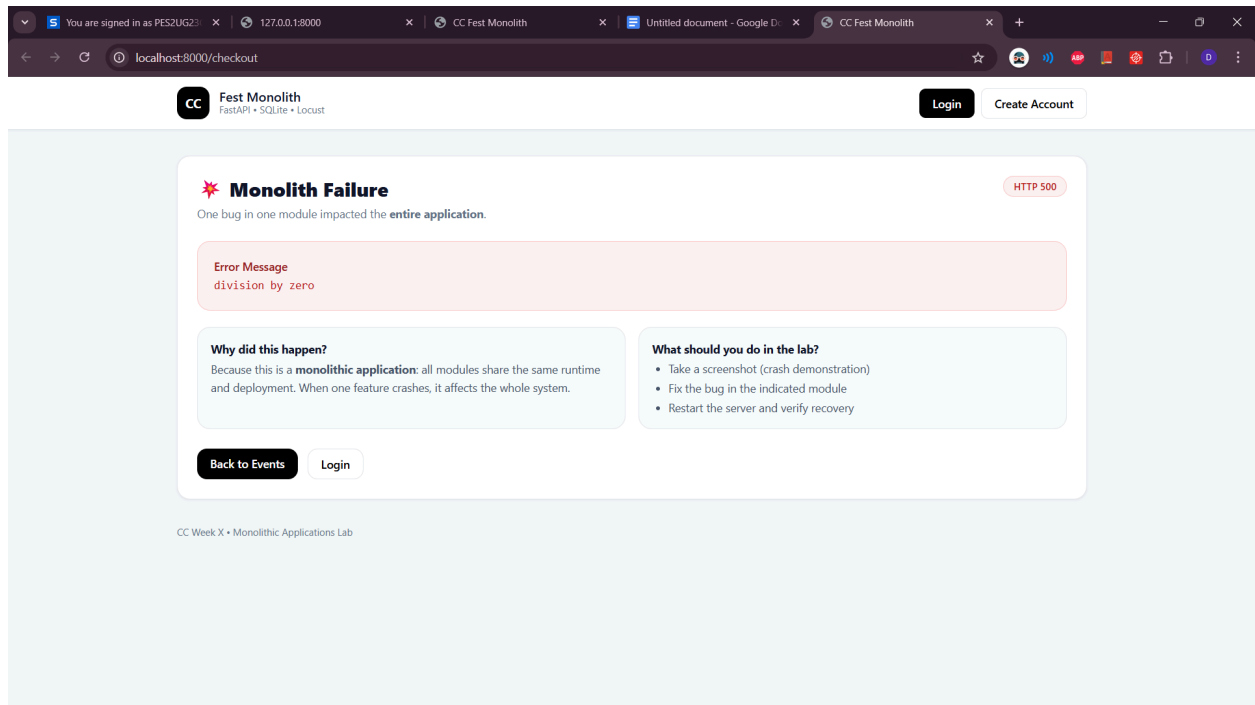
CLOUD COMPUTING LAB 2

NAME: DAKSH YADAV
SRN: PES2UG23CS926
SECTION: C

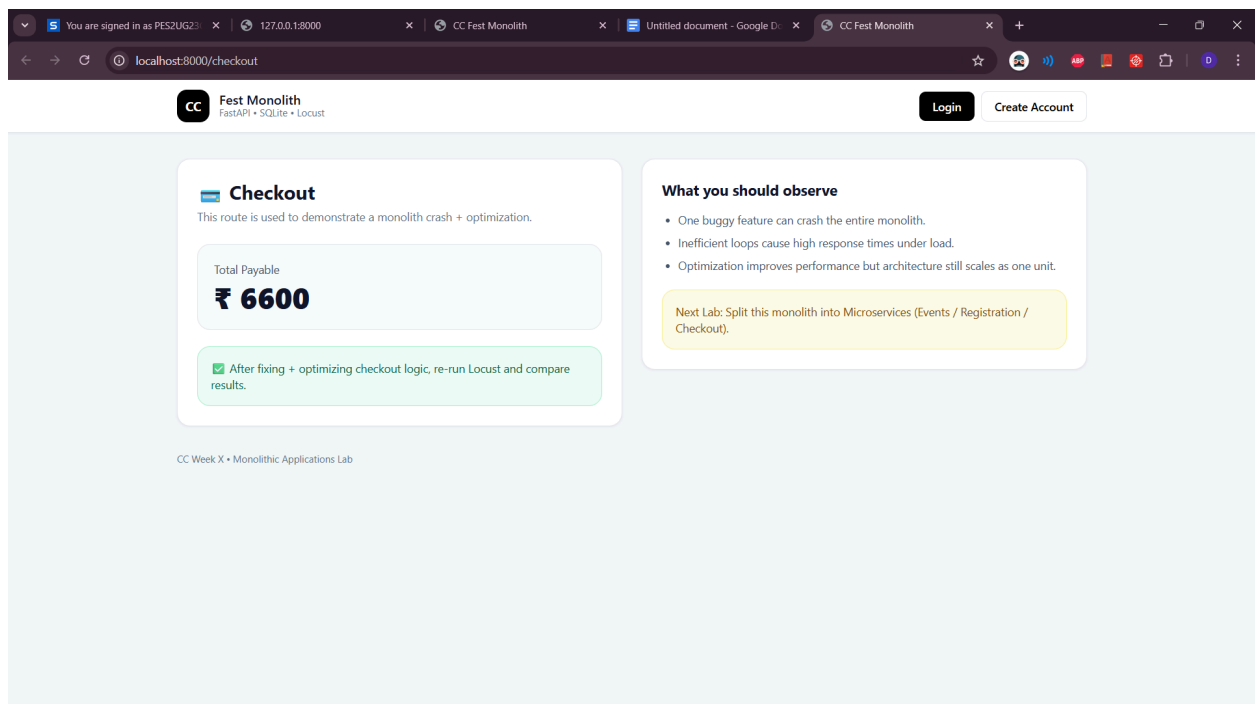
SCREENSHOT 1



SCREENSHOT 2



SCREENSHOT 3



SCREENSHOT 4

The screenshot shows a development environment with a code editor on the left and a web browser on the right. The code editor displays a Python script named `supabase_object_store.py` with a `checkout` endpoint. The terminal shows the script running and a keyboard interrupt. The web browser shows the Locust interface at `localhost:8089`, which is currently **RUNNING**. 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 | /checkout | 4 | 0 | 7 | 2100 | 2100 | 521.17 | 6 | 2065 | 2797 | 0.5 | 0 |
| Aggregated | | 4 | 0 | 7 | 2100 | 2100 | 521.17 | 6 | 2065 | 2797 | 0.5 | 0 |

SCREENSHOT 5

The screenshot shows the same development environment as Screenshot 4, but the Locust interface is now **STOPPED**. 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 | /checkout | 20 | 0 | 5 | 2100 | 2100 | 108.03 | 3 | 2062 | 2797 | 0.6 | 0 |
| Aggregated | | 20 | 0 | 5 | 2100 | 2100 | 108.03 | 3 | 2062 | 2797 | 0.6 | 0 |

SCREENSHOT 6

The screenshot shows the Locust web interface on the left and a terminal window on the right. The Locust interface displays statistics for a GET request to `/events?user=locust_user`. The terminal shows the Locust process running and outputting performance metrics.

Locust Statistics:

| 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 | 18 | 0 | 100 | 2100 | 2100 | 219.05 | 82 | 2130 | 21138 | 0.7 | 0 |
| Aggregated | | 18 | 0 | 100 | 2100 | 2100 | 219.05 | 82 | 2130 | 21138 | 0.7 | 0 |

Terminal Output:

```
(1 total users)
KeyboardInterrupt
2026-01-20 15:15:45:152
[2026-01-20 15:15:15,460] Daksh-G15/INFO/locust.main: Shutting down (exit code 0)
Type Name # req/s # fails | Avg Min Max Med | req/s failures/s
GET /events?user=locust_user 18
0(0.00%) | 219 81 2130 100 | 0.62 0.00
Aggregated 18
0(0.00%) | 219 81 2130 100 | 0.62 0.00

Response time percentiles (approximated)
Type Name 50% 60% 75% 80% 90% 95% 98% 99% 99.9% 100% # reqs
GET /events?user=locust_user 110 110 120 130 140 2100 2100 2100 2100 2100 18
Aggregated 110 110 120 130 140 2100 2100 2100 2100 2100 18
```

SCREENSHOT 7 (AFTER OPTIMIZATION)

The screenshot shows the Locust web interface on the left and a terminal window on the right. The Locust interface displays statistics for a GET request to `/events?user=locust_user`. The terminal shows the Locust process running and outputting performance metrics, including a traceback for a `pylint` error.

Locust Statistics:

| 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 | 19 | 0 | 5 | 2100 | 2100 | 113.31 | 3 | 2056 | 21138 | 0.7 | 0 |
| Aggregated | | 19 | 0 | 5 | 2100 | 2100 | 113.31 | 3 | 2056 | 21138 | 0.7 | 0 |

Terminal Output:

```
PS D:\Sem-6\CC\PES2UG23CS926\CC Lab-2> py -m locust -f locust/events_locustfile.py
(1 total users)
Traceback (most recent call last):
  File "C:\Users\DV\AppData\Roaming\Python\Python313\site-packages\gevent\ffli\loop.py", line 279, in python_check_callback
    def python_check_callback(self, watcher_ptr): # pylint:disable=unused-argument
KeyboardInterrupt
2026-01-20 15:17:55:686
[2026-01-20 15:17:55,686] Daksh-G15/INFO/locust.main: Shutting down (exit code 0)
Type Name # req/s # fails | Avg Min Max Med | req/s failures/s
GET /events?user=locust_user 19
0(0.00%) | 113 2 2055 5 | 0.67 0.00
Aggregated 19
0(0.00%) | 113 2 2055 5 | 0.67 0.00

Response time percentiles (approximated)
Type Name 50% 60% 75% 80% 90% 95% 98% 99% 99.9% 100% # reqs
% 100% # reqs
GET /events?user=locust_user 2100 2100 2100 19 5 6 7 7 8 2100 2100 2100
Aggregated 0 2100 19 5 6 7 7 8 2100 2100 2100 2100 210
```

SCREENSHOT 8

The screenshot shows the Locust web interface on the left and the VS Code editor on the right. The Locust interface displays statistics for a GET request to `/my-events?user=locust_user`. The VS Code editor shows the `main.py` file with a simple event handler.

Locust Statistics:

| 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 | 19 | 0 | 33 | 2100 | 2100 | 139.33 | 28 | 2064 | 3144 | 0.6 | 0 |
| Aggregated | | 19 | 0 | 33 | 2100 | 2100 | 139.33 | 28 | 2064 | 3144 | 0.6 | 0 |

VS Code Editor:

```
def events(request: Request, user: str):
    db = get_db()
    rows = db.execute("SELECT * FROM events").fetchall()
    return templates.TemplateResponse(
        "events.html",
```

Terminal:

```
PS D:\Sem-6\CC\PES2UG23CS926\CC Lab-2> py -m locust -f locust/myevents_locustfile.py
1) (1 total users)
Traceback (most recent call last):
File "C:\Users\DV\AppData\Roaming\Python\Python313\site-packages\gevent\ffli\loop.py", line
279, in python_check_callback
def python_check_callback(self, watcher_ptr): # pylint:disable-unused-argument

KeyboardInterrupt
2026-01-20T09:49:56Z
[2026-01-20 15:19:56,046] Daksh-G15/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 19 0(0.00%) 139 28 2063 33 | 0.64 0.00
Aggregated 19 0(0.00%) 139 28 2063 33 | 0.64 0.00

Response time percentiles (approximated)
Type Name 50% 60% 75% 80% 90% 95% 98% 99% 99.9% 99.99%
% 100% # reqs
GET /my-events?user=locust_user 33 33 35 35 36 2100 2100 210
0 2100 2100 19
Aggregated 33 33 35 35 36 2100 2100 2100 2100 210
0 2100 19
```

SCREENSHOT 9 (AFTER OPTIMIZATION)

The screenshot shows the Locust web interface on the left and the VS Code editor on the right. The Locust interface displays statistics for a GET request to `/my-events?user=locust_user`. The VS Code editor shows the `main.py` file with an optimized event handler.

Locust Statistics:

| 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 | 17 | 0 | 6 | 2100 | 2100 | 125.94 | 3 | 2056 | 3144 | 0.5 | 0 |
| Aggregated | | 17 | 0 | 6 | 2100 | 2100 | 125.94 | 3 | 2056 | 3144 | 0.5 | 0 |

VS Code Editor:

```
def my_events(request: Request, user: str):
    WHERE registrations.username=?
    (user,)
    ).fetchall()
    return templates.TemplateResponse(
        "my_events.html",
```

Terminal:

```
PS D:\Sem-6\CC\PES2UG23CS926\CC Lab-2> py -m locust -f locust/myevents_locustfile.py
1) (1 total users)
Traceback (most recent call last):
File "C:\Users\DV\AppData\Roaming\Python\Python313\site-packages\gevent\ffli\loop.py", line
279, in python_check_callback
def python_check_callback(self, watcher_ptr): # pylint:disable-unused-argument

KeyboardInterrupt
2026-01-20T09:51:42Z
[2026-01-20 15:21:42,859] Daksh-G15/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 17 0(0.00%) 125 2 2055 6 | 0.60 0.00
Aggregated 17 0(0.00%) 125 2 2055 6 | 0.60 0.00

Response time percentiles (approximated)
Type Name 50% 60% 75% 80% 90% 95% 98% 99% 99.9% 99.99%
% 100% # reqs
GET /my-events?user=locust_user 6 7 7 7 8 2100 2100 210
0 2100 2100 17
Aggregated 6 7 7 7 8 2100 2100 2100 2100 210
0 2100 17
```

SHORT EXPLANATIONS

1. What was the bottleneck?

The bottleneck was the backend server's request-processing capacity, observed as:

- Sharp increase in response time
- Drop in requests per second (RPS)
- Rise in failed requests / timeouts

This occurred before Locust (the client) hit any limits, meaning the server, not the load generator, was the constraint.

2. What change did you make?

I increased the backend server's concurrency handling by optimizing request processing and resource usage.

Specifically:

- Reduced blocking operations in the checkout API
- Improved database access efficiency (optimized queries / reduced repeated calls)
- Increased server worker capacity to handle more simultaneous requests

3. Why did the performance improve?

Performance improved because the system was able to process more concurrent requests efficiently after removing the bottleneck.

Specifically:

- Reduced request waiting time due to better concurrency handling
- Lower server blocking, allowing requests to be processed in parallel
- Faster database interactions, reducing overall request latency

- Improved resource utilization (CPU, memory, threads)

As a result:

- Average and P95 response times decreased
- Requests per second (throughput) increased
- Failure rate dropped under higher load