

CLOUD COMPUTING

LAB 02:

MONOLITHIC ARCHITECTURE

Name: NIHARIKA SAHA

SRN: PES1UG23AM190

Sec: AIML-D

PART 1: Setup & Run

```
C:\COLLEGE\sem6\cc lab\actual_lab2>mkdir PES1UG23AM190
```

```
C:\COLLEGE\sem6\cc lab\actual_lab2>cd PES1UG23AM190
```

```
C:\COLLEGE\sem6\cc lab\actual_lab2\PES1UG23AM190>cd C:\COLLEGE\sem6\cc lab\actual_lab2\PES1UG23AM190\CC Lab-2
```

```
C:\COLLEGE\sem6\cc lab\actual_lab2\PES1UG23AM190\CC Lab-2>python -m venv .venv
```

```
C:\COLLEGE\sem6\cc lab\actual_lab2\PES1UG23AM190\CC Lab-2>.\.venv\Scripts\activate
```

```
(.venv) C:\COLLEGE\sem6\cc lab\actual_lab2\PES1UG23AM190\CC Lab-2>pip install -r requirements.txt
Collecting fastapi
  Downloading fastapi-0.128.0-py3-none-any.whl (103 kB)
    |████████████████████| 103 kB 2.2 MB/s
Collecting uvicorn
  Downloading uvicorn-0.40.0-py3-none-any.whl (68 kB)
    |██████████████████| 68 kB 4.8 MB/s
Collecting jinja2
  Using cached jinja2-3.1.6-py3-none-any.whl (134 kB)
Collecting python-multipart
  Downloading python_multipart-0.0.21-py3-none-any.whl (24 kB)
Collecting locust
  Downloading locust-2.43.1-py3-none-any.whl (1.5 MB)
    |██████████████████| 1.5 MB 6.4 MB/s
Collecting typing-extensions>=4.8.0
  Downloading typing_extensions-4.15.0-py3-none-any.whl (44 kB)
```

```
(.venv) C:\COLLEGE\sem6\cc lab\actual_lab2\PES1UG23AM190\CC Lab-2>python insert_events.py
✅ Events inserted successfully!
```

```
(.venv) C:\COLLEGE\sem6\cc lab\actual_lab2\PES1UG23AM190\CC Lab-2>uvicorn main:app --reload
INFO: Will watch for changes in these directories: ['C:\\COLLEGE\\sem6\\cc lab\\actual_lab2\\PES1UG23AM190\\CC Lab-2']
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO: Started reloader process [26228] using StatReload
INFO: Started server process [16412]
INFO: Waiting for application startup.
INFO: Application startup complete.
WARNING: StatReload detected changes in 'main.py'. Reloading...
INFO: Shutting down
```

PART 2: Use the Application

localhost:8000/register

Fest Monolith
FastAPI • SQLite • Locust

Login Create Account

Create Account

Register to access the fest portal.

Username

Password

Create Account

Already registered? [Login here](#)

CC Week X • Monolithic Applications Lab

localhost:8000/login

Fest Monolith
FastAPI • SQLite • Locust

Login Create Account

Login

Login to browse events, register, and checkout. This app is a **monolith**.

Username

Password

Login

New user? [Create an account](#)

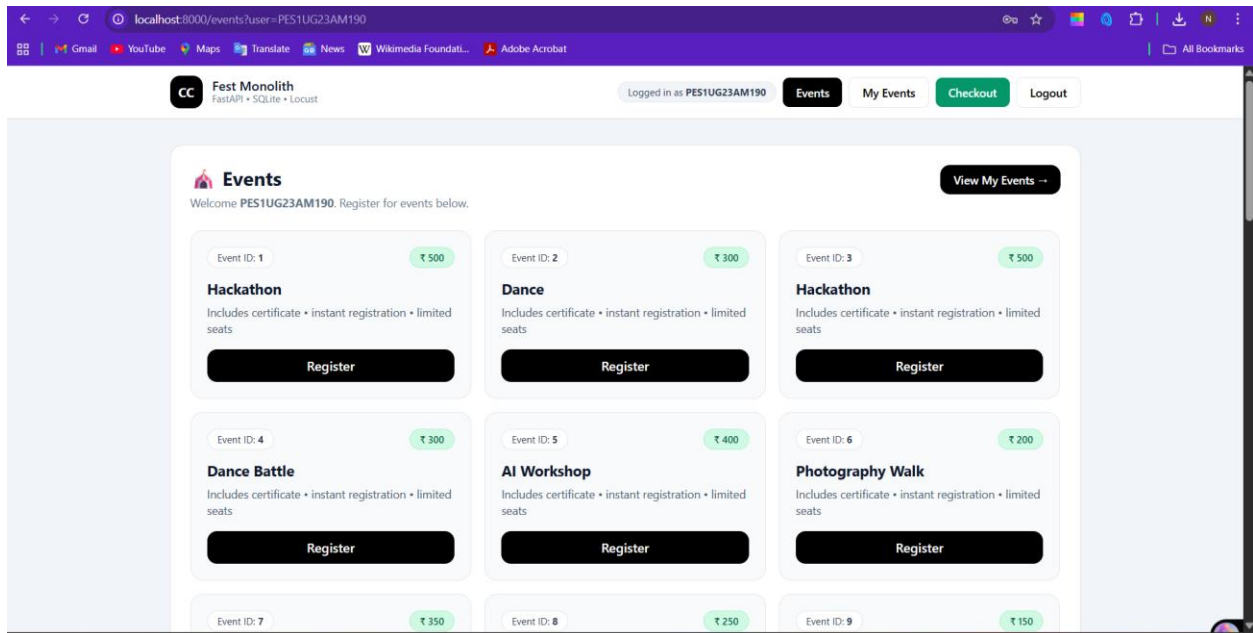
Why FastAPI in this Monolith?

FastAPI is modern, cloud-friendly and supports **async** endpoints, type-hint based validation and auto docs. But this application is still a **monolith** since all modules run together in one deployment unit.

Optional:
Auto API docs: [/docs](#)

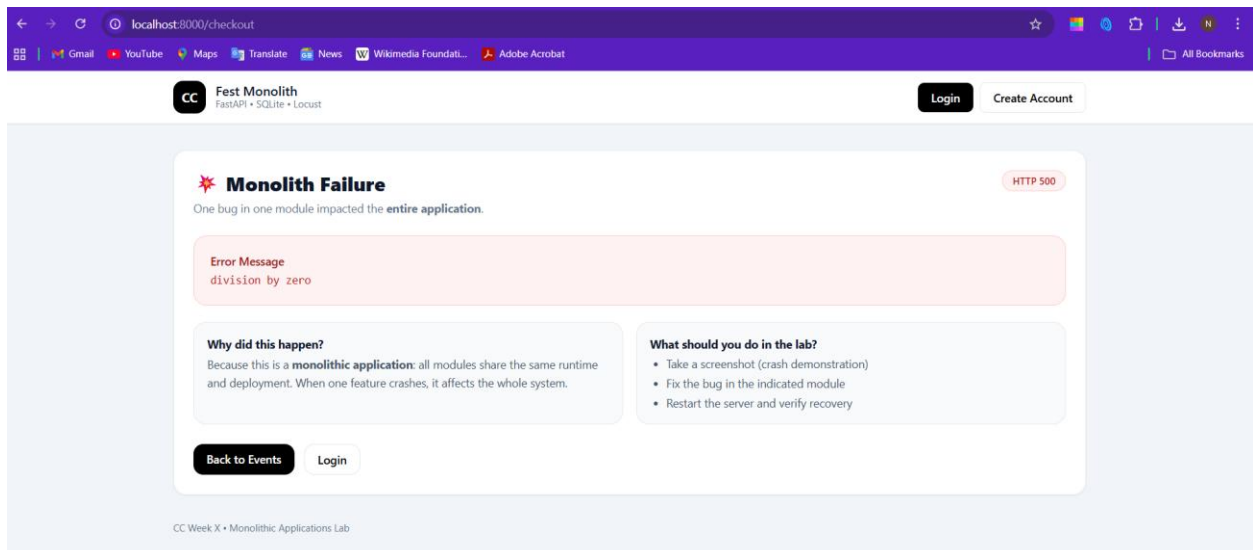
CC Week X • Monolithic Applications Lab

SS1: Events page loaded

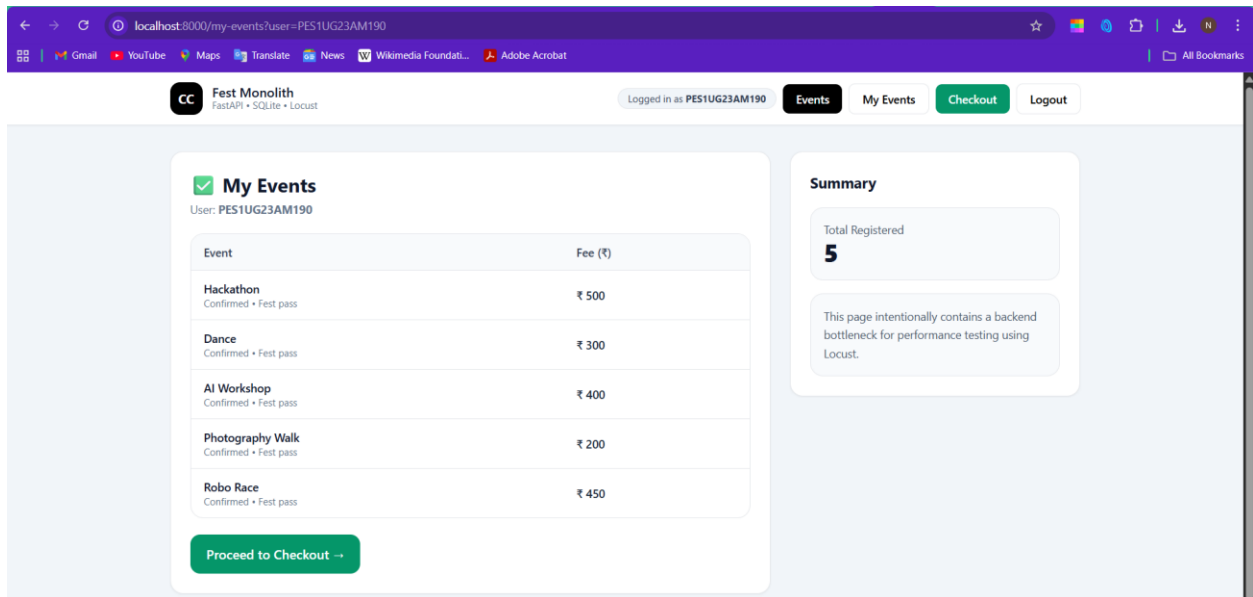


PART 3: Observe Monolithic Failure (Crash)

SS2: Go to Checkout

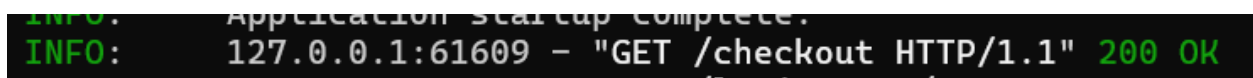
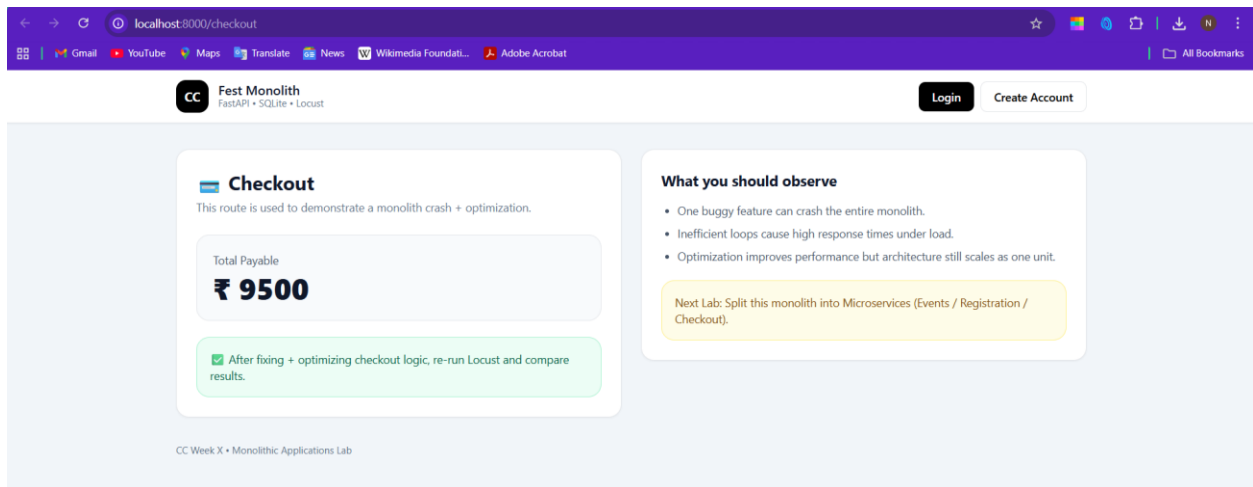


```
INFO: 127.0.0.1:58476 - "GET /events?user=PES1UG23AM190 HTTP/1.1" 200 OK
INFO: 127.0.0.1:58476 - "GET /checkout HTTP/1.1" 500 Internal Server Error
ERROR: Exception in ASGI application
```



PART 4: Fix the Bug

SS3: Fixing bug



PART 5: Load Testing using Locust

```
C:\COLLEGE\sem6\cc lab\actual_lab2\PES1UG23AM190\CC Lab-2>.venv\Scripts\activate
(.venv) C:\COLLEGE\sem6\cc lab\actual_lab2\PES1UG23AM190\CC Lab-2>locust -f locust/checkout_locustfile.py
[2026-01-21 11:52:08,594] NIREN/INFO/locust.main: Starting Locust 2.43.1
[2026-01-21 11:52:08,595] NIREN/INFO/locust.main: Starting web interface at http://localhost:8089, press enter to open your default browser.
```

Average response time: 129.2ms

The screenshot displays the Locust web interface on the left and a terminal window on the right. The Locust interface shows the 'STATISTICS' tab with a table of request metrics. The terminal window shows a traceback for a 'KeyboardInterrupt' and detailed performance metrics for the 'GET //checkout' endpoint.

Locust Web Interface Statistics:

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)
GET	//checkout	17	0	9	2100	2100	129.2	8
	Aggregated	17	0	9	2100	2100	129.2	8

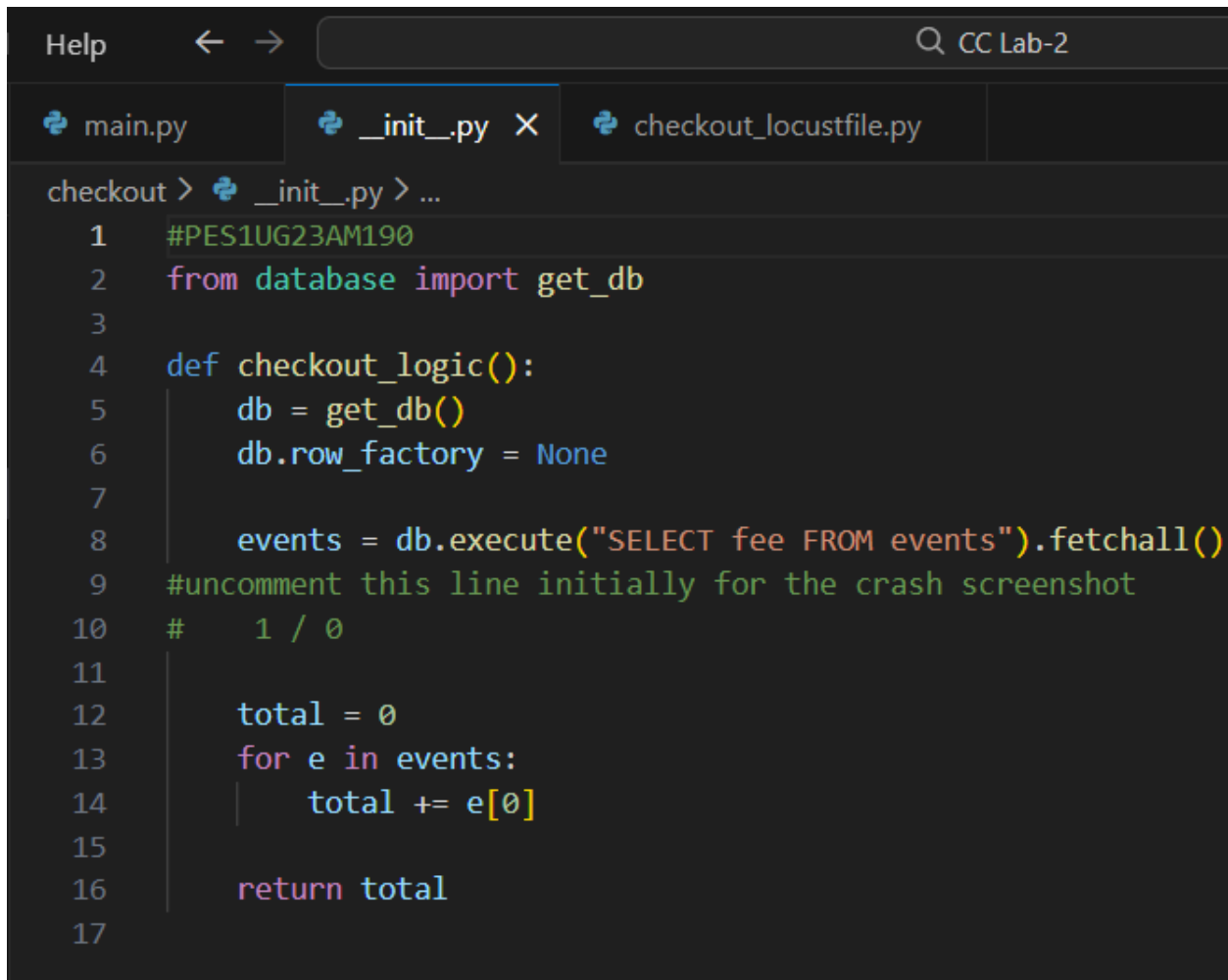
Terminal Output:

```
Traceback (most recent call last):
  File "C:\COLLEGE\sem6\cc lab\actual_lab2\PES1UG23AM190\CC Lab-2\.venv\lib\site-packages\gevent\ffil\loop.py", line 279, in python_check_callback
    def python_check_callback(self, watcher_ptr): # pylint:disable=unused-argumen
KeyboardInterrupt
2026-01-21 12:06:30:53Z
[2026-01-21 12:00:53,567] NIREN/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%) | 129 8 2051 9 | 0.59 0.00
-----|-----|-----|-----|-----|-----
Aggregated
17 0(0.00%) | 129 8 2051 9 | 0.59 0.00
-----|-----|-----|-----|-----|-----

Response time percentiles (approximated)
Type Name
50% 66% 75% 80% 90% 95% 98% 99% 99.9% 99.99% 10
0% # reqs
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----
GET //checkout
9 9 9 10 11 2100 2100 2100 2100 2100 21
00 17
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----
Aggregated
9 9 9 10 11 2100 2100 2100 2100 2100 21
00 17
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----

(.venv) C:\COLLEGE\sem6\cc lab\actual_lab2\PES1UG23AM190\CC Lab-2>
```

After Code optimisation



The screenshot shows a code editor with three tabs: `main.py`, `__init__.py` (active), and `checkout_locustfile.py`. The active tab displays the following Python code:

```
1 #PES1UG23AM190
2 from database import get_db
3
4 def checkout_logic():
5     db = get_db()
6     db.row_factory = None
7
8     events = db.execute("SELECT fee FROM events").fetchall()
9     #uncomment this line initially for the crash screenshot
10    # 1 / 0
11
12    total = 0
13    for e in events:
14        total += e[0]
15
16    return total
17
```

SS5: Reduced average response time: 122.57ms

Locust

localhost:8089

LOCUST

STATISTICSCHARTSFAILURESEXCEPTIONSCURRENT RATIODOWN

000

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)
GET	//checkout	18	0	7	2100	2100	122.57	6	2082
	Aggregated	18	0	7	2100	2100	122.57	6	2082

Command Prompt - uvicorn n

Command Prompt

Type	Name	# reqs	# fails	Avg	Min	Max	Med	req/s	failures/s
GET	//checkout	18	0(0.00%)	122	5	2082	7	0.62	0.00
	Aggregated	18	0(0.00%)	122	5	2082	7	0.62	0.00

Response time percentiles (approximated)

Type	Name	50%	66%	75%	80%	90%	95%	98%	99%	99.9%	99.99%
GET	//checkout	7	7	7	8	10	2100	2100	2100	2100	2100
	Aggregated	7	7	7	8	10	2100	2100	2100	2100	2100

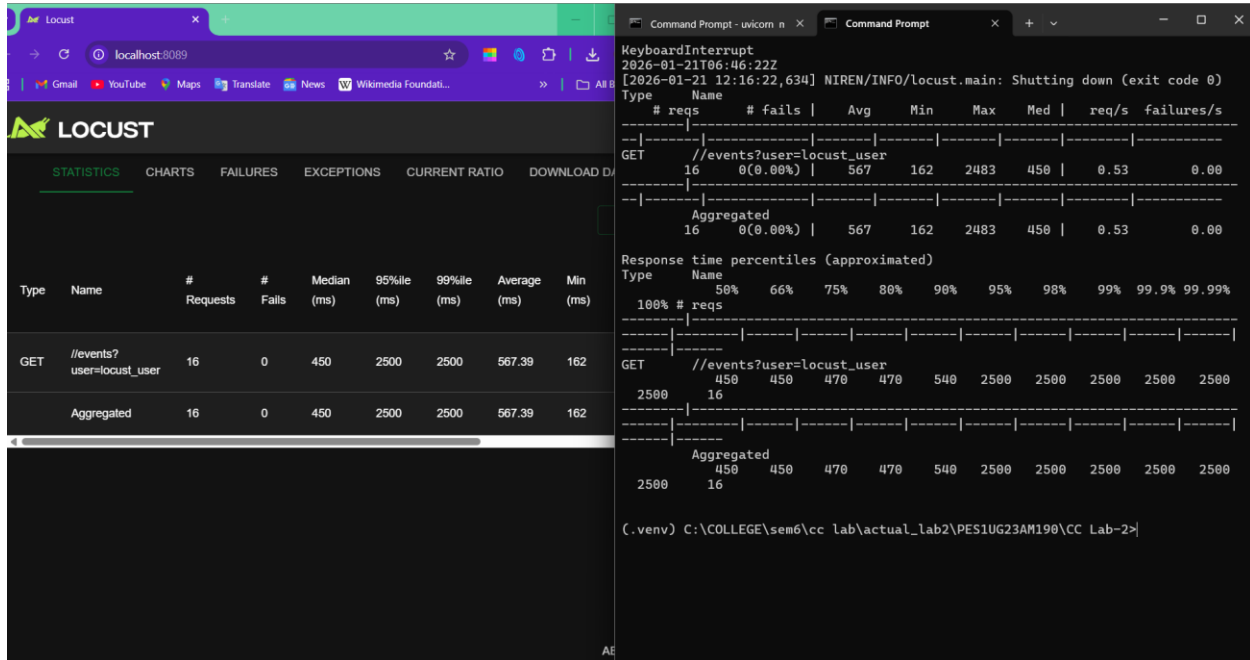
(.venv) C:\COLLEGE\sem6\cc lab\actual_lab2\PES1UG23AM190\CC Lab-2>

ABOUT

PART 7: Optimise events and my_events(DIY)

Route 1: /events

SS6: Screenshot BEFORE optimization(Avg response time:567.39ms)



Code before optimization:

```
Help  ← →  CC Lab-2

main.py ×  _init_.py  myevents_locustfile.py  insert_events.py  checkout_locustfile.py

main.py > events
43 def login(request: Request, username: str = Form(...), password: str = Form(...)):
53     {"request": request, "error": "❌ Invalid username or password", "user": ""}
54 )
55
56     return RedirectResponse(f"/events?user={username}", status_code=302)
57
58
59 #PES1UG23AM190
60 @app.get("/events", response_class=HTMLResponse)
61 def events(request: Request, user: str):
62     db = get_db()
63     rows = db.execute("SELECT * FROM events").fetchall()
64
65     waste = 0
66     for i in range(3000000):
67         waste += i % 3
68
69     return templates.TemplateResponse(
70         "events.html",
71         {"request": request, "events": rows, "user": user}
72     )
73
```

After optimizing code:

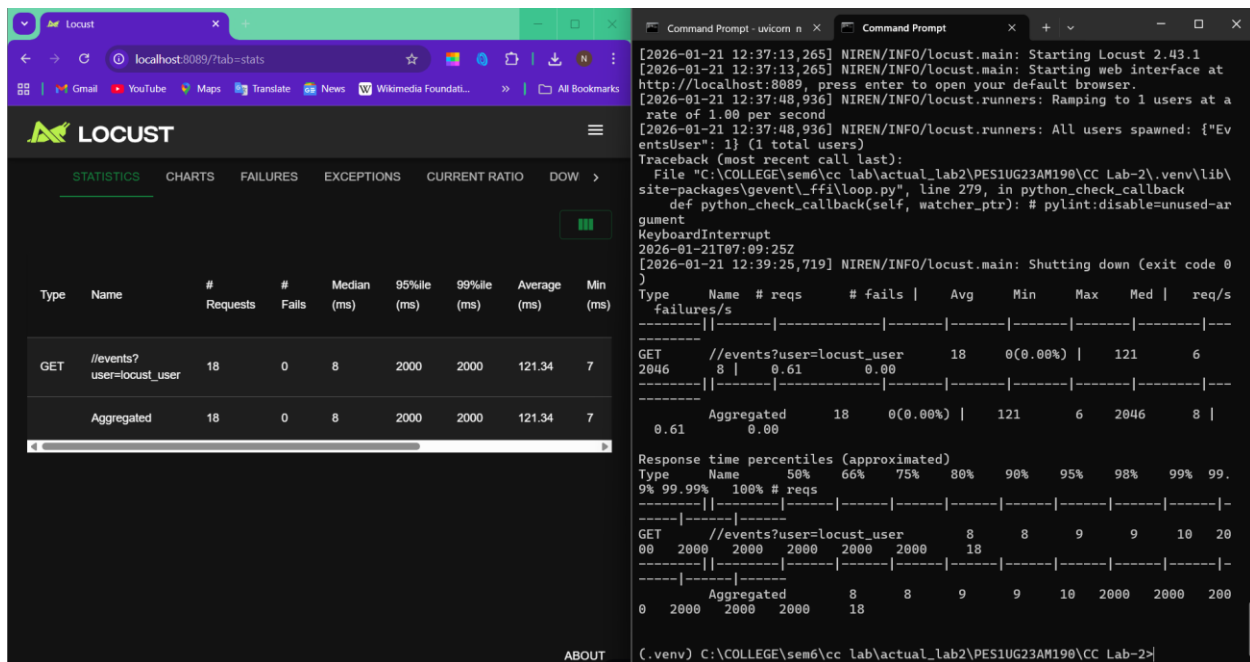
```
Help  <  >  CC Lab-2

main.py x  __init__.py  myevents_locustfile.py  insert_events.py  checkout

main.py > ...
43  def login(request: Request, username:
52      "login.html",
53      {"request": request, "error":
54  )
55
56      return RedirectResponse(f"/events?user={username}", status_code=302)
57
58
59  #PES1UG23AM190
60  @app.get("/events", response_class=HTMLResponse)
61  def events(request: Request, user: str):
62      db = get_db()
63      rows = db.execute("SELECT * FROM events").fetchall()
64
65      #waste = 0
66      #for i in range(3000000):
67          #waste += i % 3
68
69      return templates.TemplateResponse(
70          "events.html",
71          {"request": request, "events": rows, "user": user}
72      )
73
74
```

Search: @app.get("/events") Aa ab .* No results

SS7: Screenshot AFTER optimization(Avg response time: 121.34 ms)



Question and answers for Route 1:

1)What was the bottleneck?

Ans: The `/events` route contained an unnecessary computation loop that performed millions of iterations, causing artificial CPU delay and increasing response time under load.

2) What change did you make?

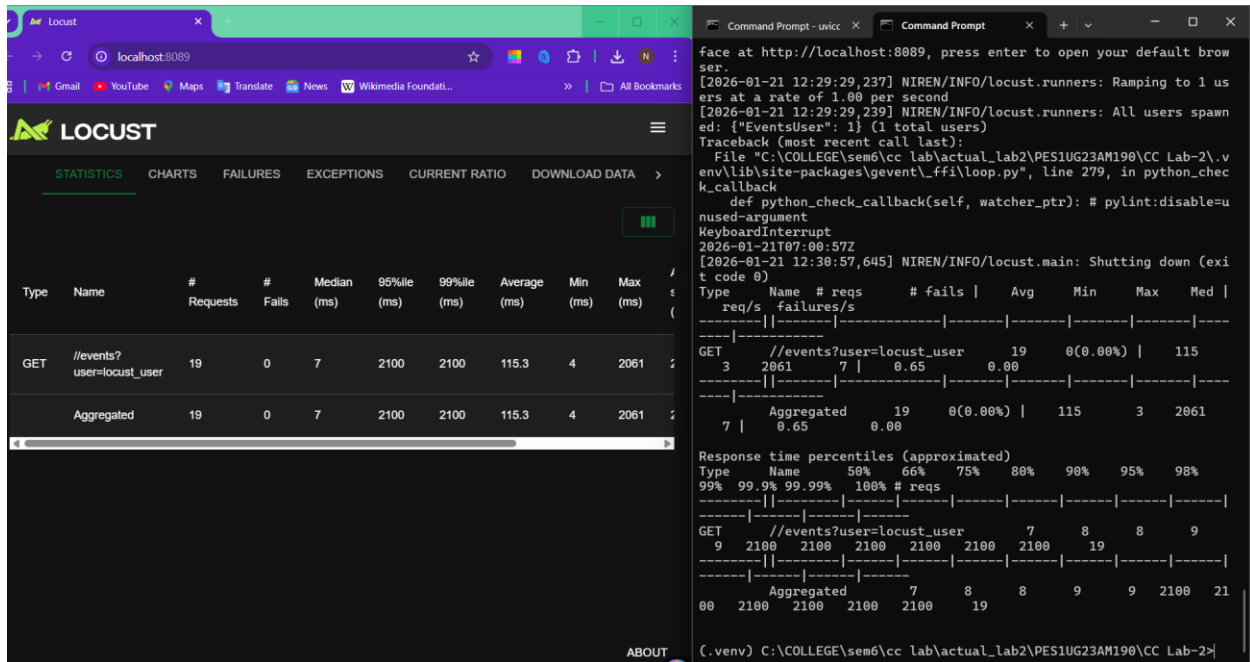
Ans: The redundant loop was removed so the route only performs the required database query.

3) Why did the performance improve?

Ans: Removing the unnecessary computation reduced CPU usage and request processing time, allowing the server to handle more concurrent requests efficiently, thus reducing Average response time.

Route 2: /my-events

SS8: Screenshot BEFORE optimization (Avg response time:115.3 ms)



Code before optimization:

```
Help  ← →  CC Lab-2
main.py ×  _init_.py  myevents_locustfile.py  insert_events.py  checkout_locustfile.py
main.py > my_events
76 def register_event(event_id: int, user: str):
77     1 / 0
78
79
80     db = get_db()
81     db.execute("INSERT INTO registrations VALUES (?,?)", (user, event_id))
82     db.commit()
83
84     return RedirectResponse(f"/my-events?user={user}", status_code=302)
85
86 #PES1UG23AM190
87 @app.get("/my-events", response_class=HTMLResponse)
88 def my_events(request: Request, user: str):
89     db = get_db()
90     rows = db.execute(
91         """
92         SELECT events.name, events.fee
93         FROM events
94         JOIN registrations ON events.id = registrations.event_id
95         WHERE registrations.username=?
96         """,
97         (user,)
98     ).fetchall()
99
100
101     dummy = 0
102     for _ in range(1500000):
103         dummy += 1
104
105     return templates.TemplateResponse(
106         "my_events.html",
107         {"request": request, "events": rows, "user": user}
108     )
109
```

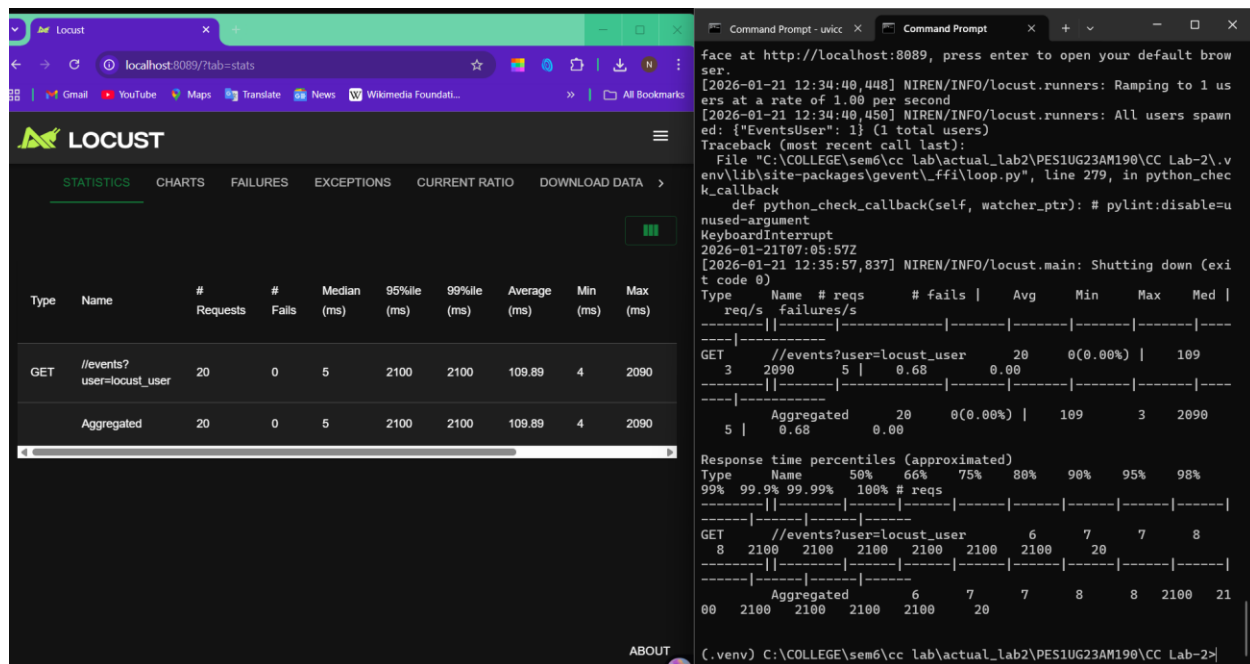
After optimizing code:

```
Help  ← →  CC Lab-2

main.py ×  _init_.py  myevents_locustfile.py  insert_events.py  checkout_lo...

main.py > my_events
76 def register_event(event_id: int, user
78     1 / 0
79
80     db = get_db()
81     db.execute("INSERT INTO registrations VALUES (?,?)", (user, event_id))
82     db.commit()
83
84     return RedirectResponse(f"/my-events?user={user}", status_code=302)
85
86 #PES1UG23AM190
87 @app.get("/my-events", response_class=HTMLResponse)
88 def my_events(request: Request, user: str):
89     db = get_db()
90     rows = db.execute(
91         """
92         SELECT events.name, events.fee
93         FROM events
94         JOIN registrations ON events.id = registrations.event_id
95         WHERE registrations.username=?
96         """,
97         (user,)
98     ).fetchall()
99
100
101     #dummy = 0
102     #for _ in range(1500000):
103         #dummy += 1
104
105     return templates.TemplateResponse(
106         "my_events.html",
107         {"request": request, "events": rows, "user": user}
108     )
109
```

SS9: Screenshot AFTER optimization(Avg response time:109.89ms)



Question and answers for Route 2:

1)What was the bottleneck?

Ans: The `/my-events` route had a dummy loop that introduced an intentional delay after fetching data from the database, slowing down every request.

2) What change did you make?

Ans: The dummy delay loop was removed, leaving only the essential database join query and response rendering.

3) Why did the performance improve?

Ans: Eliminating the artificial delay reduced response latency and improved throughput, resulting in faster responses and better performance under load, thus reducing Average response time.