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
```

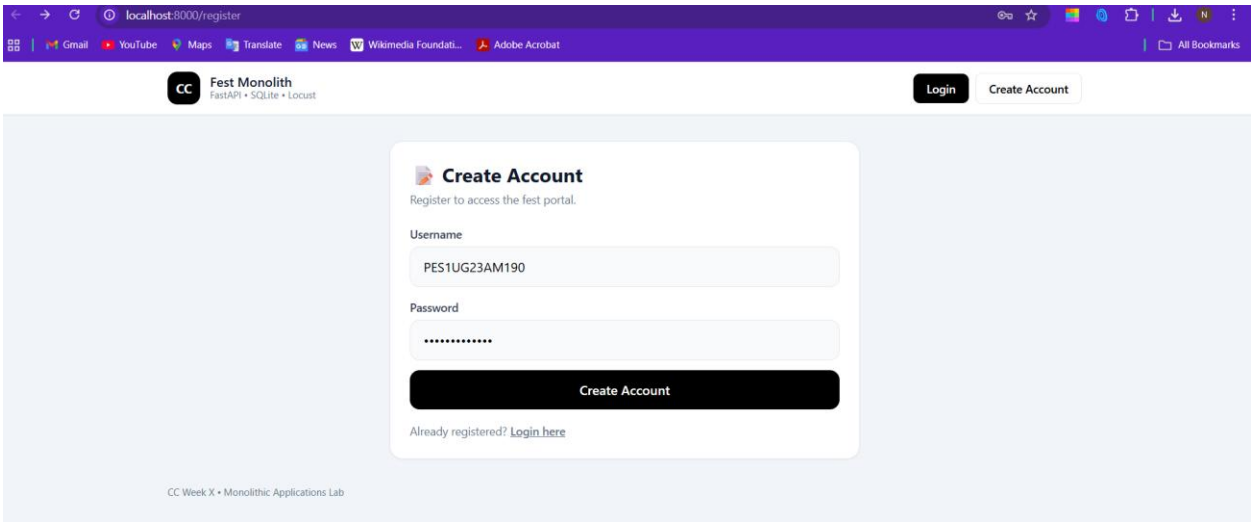
```
C:\COLLEGE\sem6\cc lab\actual_lab2\PES1UG23AM190\CC Lab-2>python -m venv .venv

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

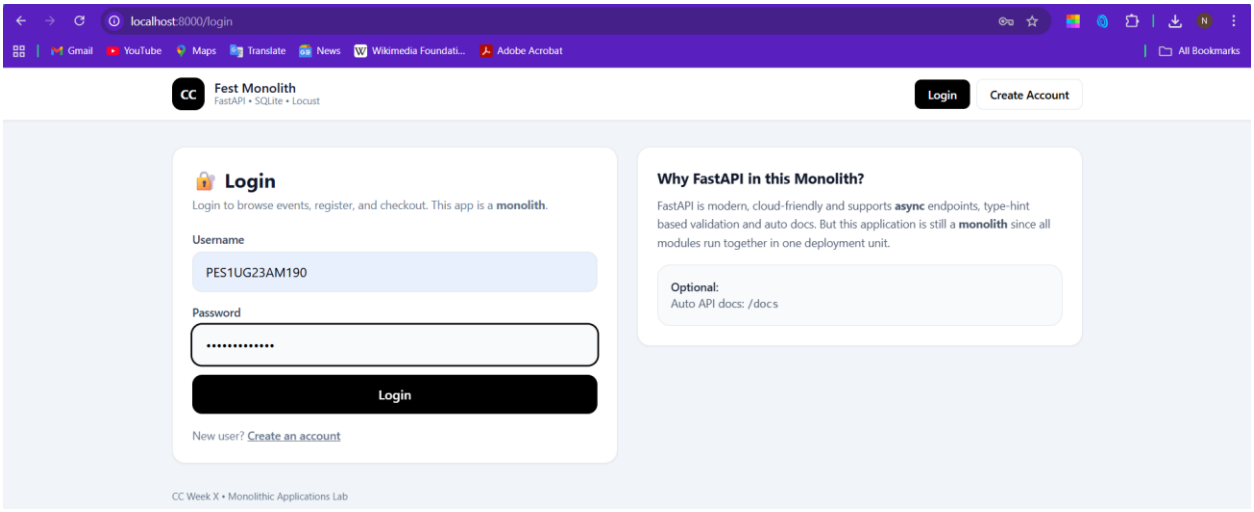
```
(.venv) C:\COLLEGE\sem6\cc lab\actual_lab2\PES1UG23AM190\CC Lab-2>pip install -r requirements.txt
Collecting fastapi
  Downloading fastapi-0.128.0-py3-none-any.whl (103 kB)
     |████████████████████████████████| 103 kB 2.2 MB/s
Collecting uvicorn
  Downloading uvicorn-0.40.0-py3-none-any.whl (68 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!
```

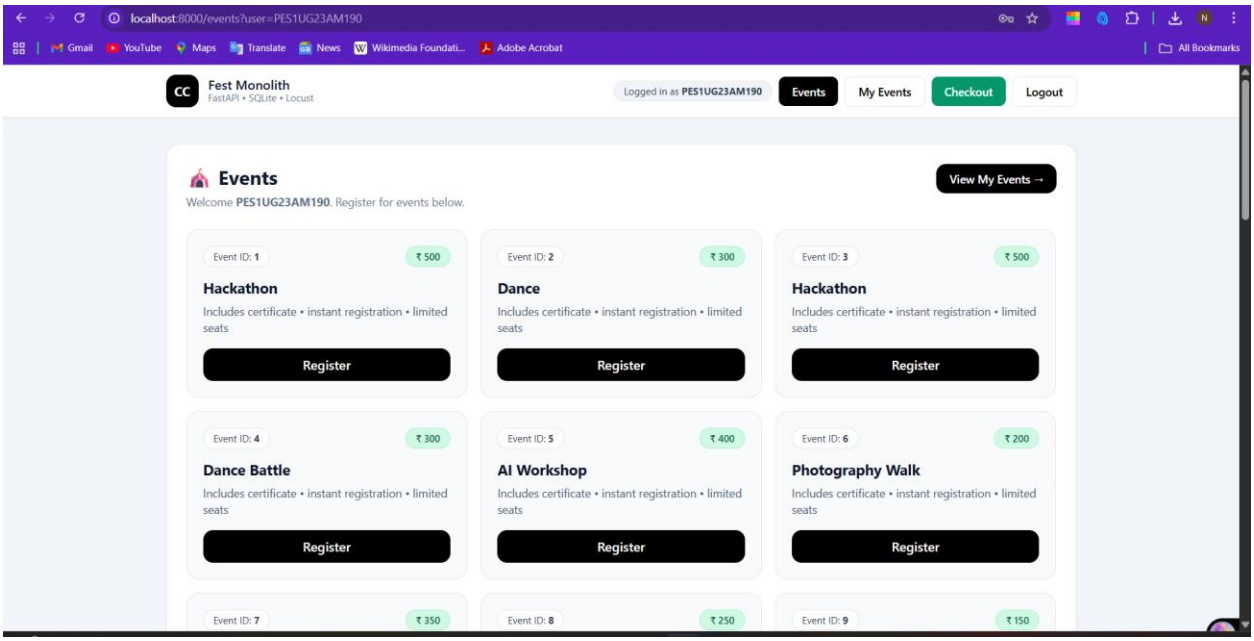
```
(.venv) C:\COLLEGE\sem6\cc lab\actual_lab2\PES1UG23AM190\CC Lab-2>uvicorn main:app --reload
INFO:     Will watch for changes in these directories: ['C:\\COLLEGE\\sem6\\cc lab\\actual_lab2\\PES1UG23AM190\\CC Lab-2']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     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



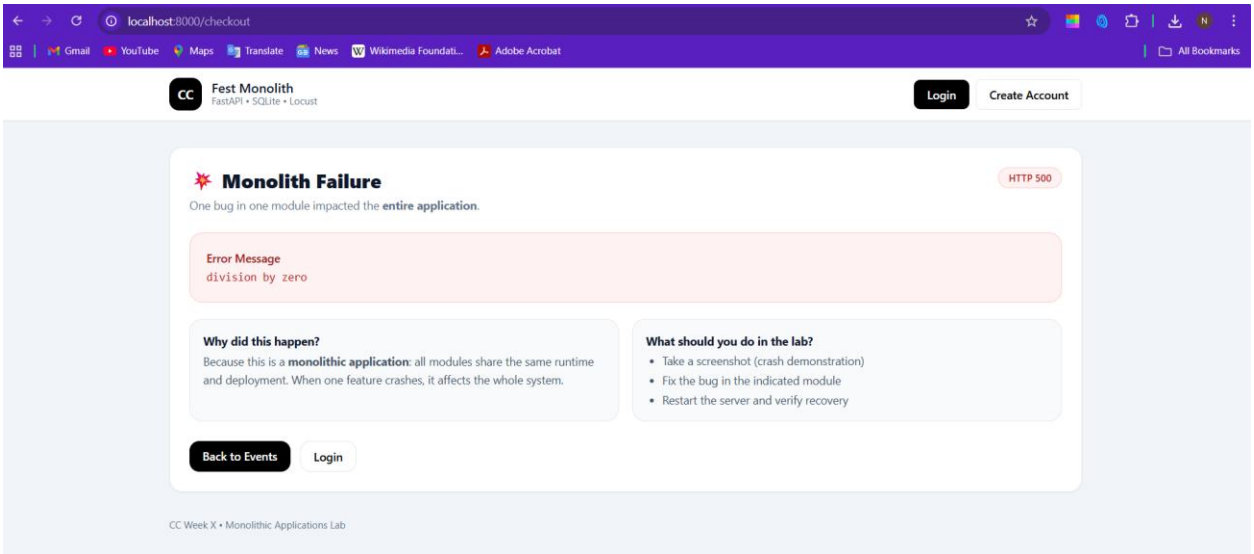


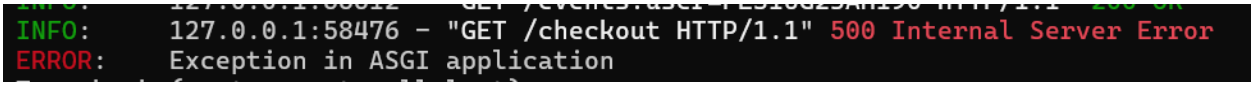
SS1: Events page loaded

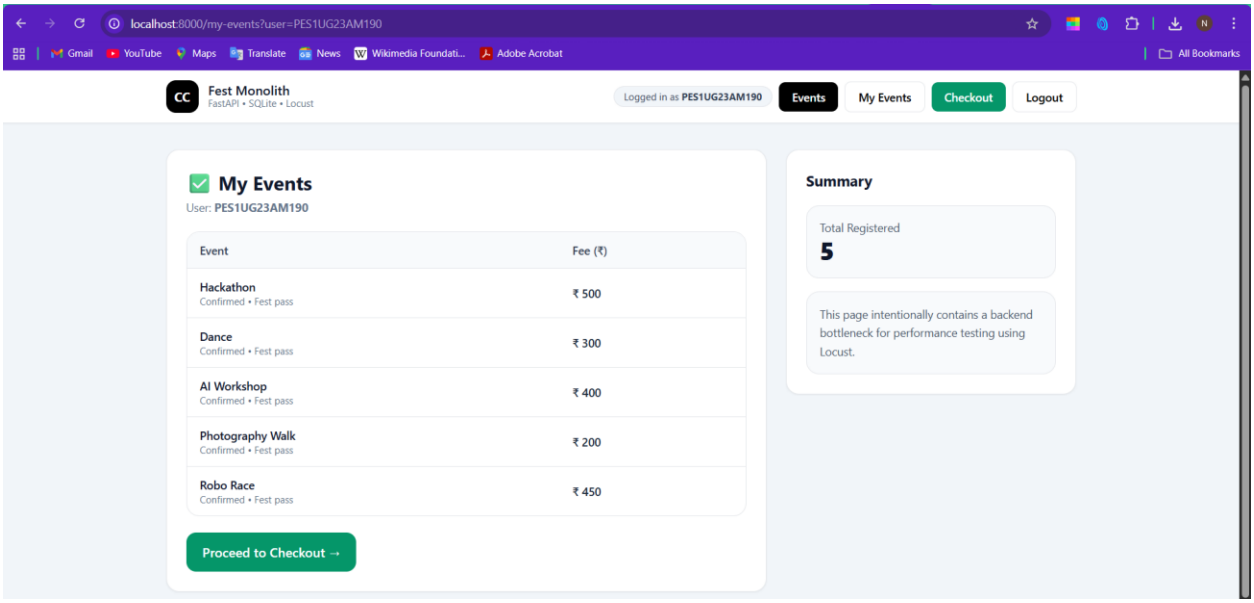


PART 3: Observe Monolithic Failure (Crash)

SS2: Go to Checkout

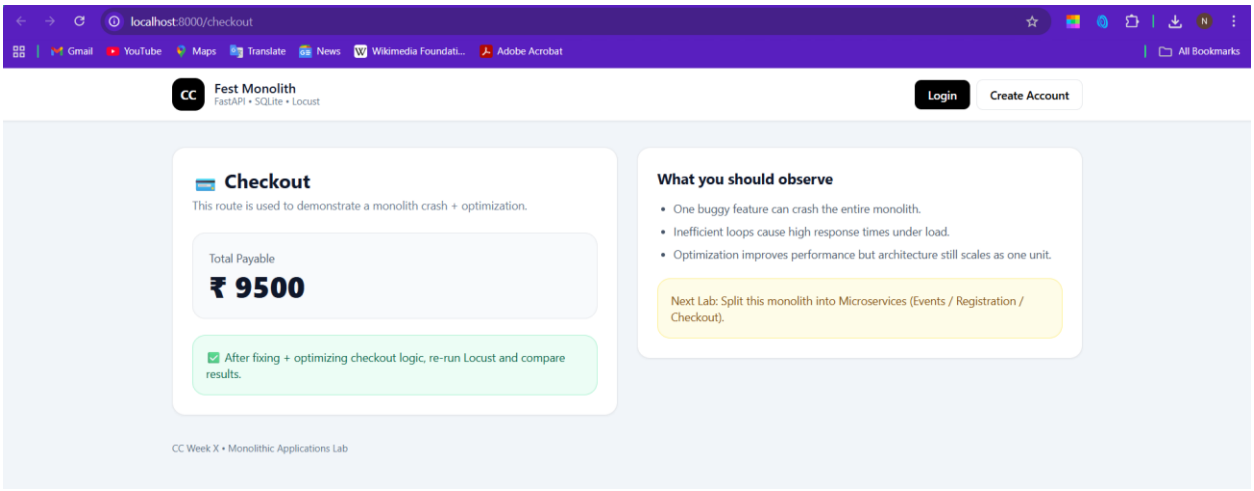






PART 4: Fix the Bug

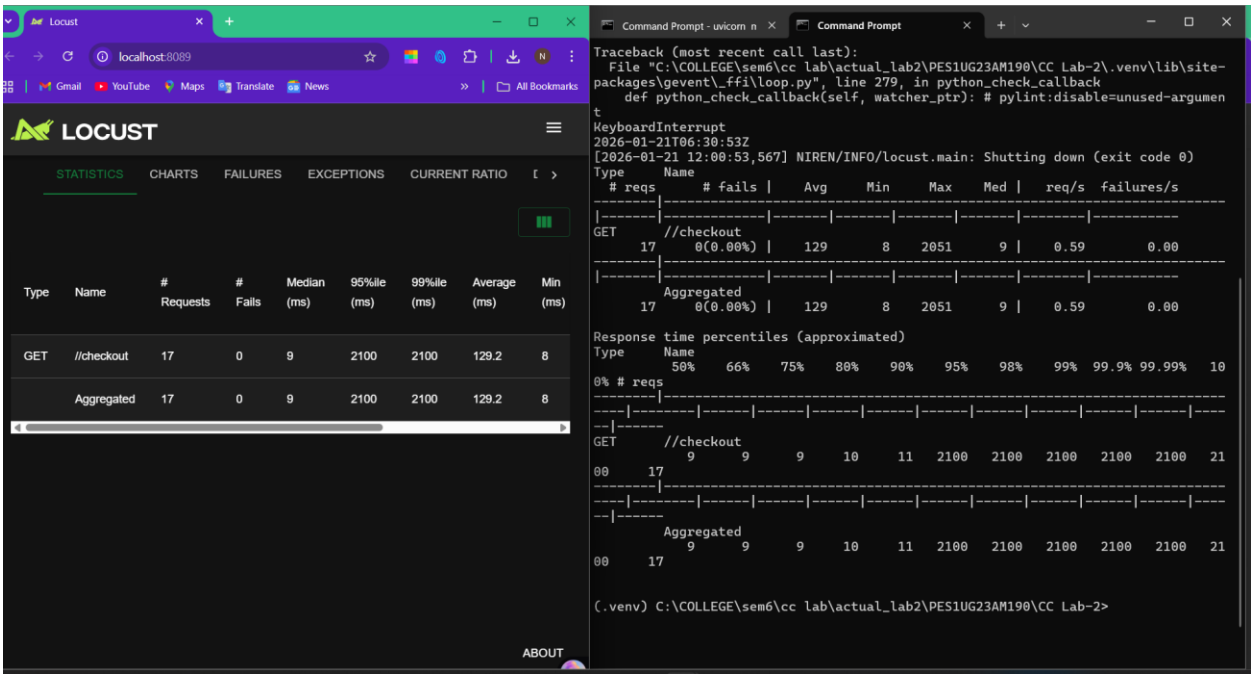
SS3: Fixing bug



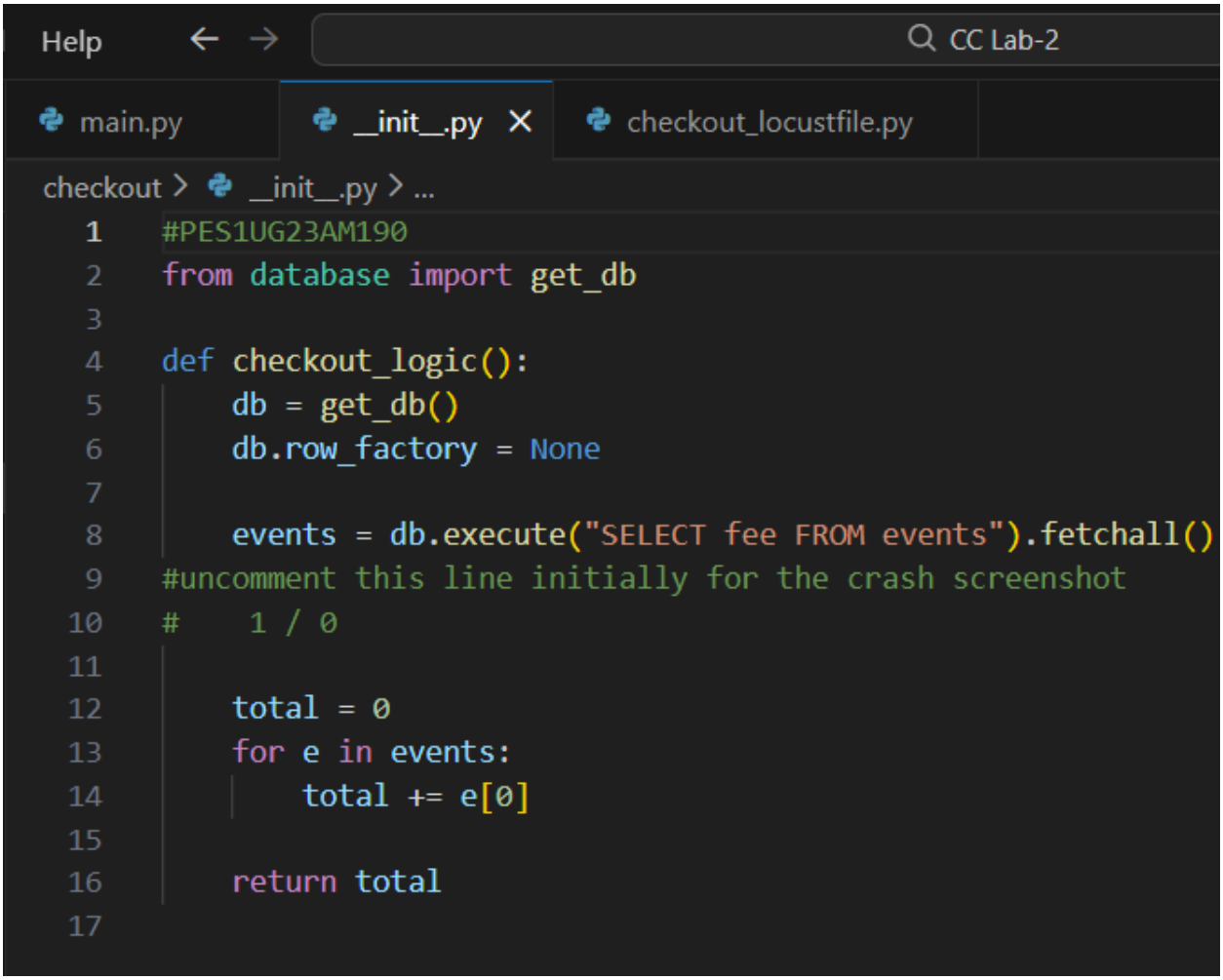
PART 5: Load Testing using Locust

```
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.
```

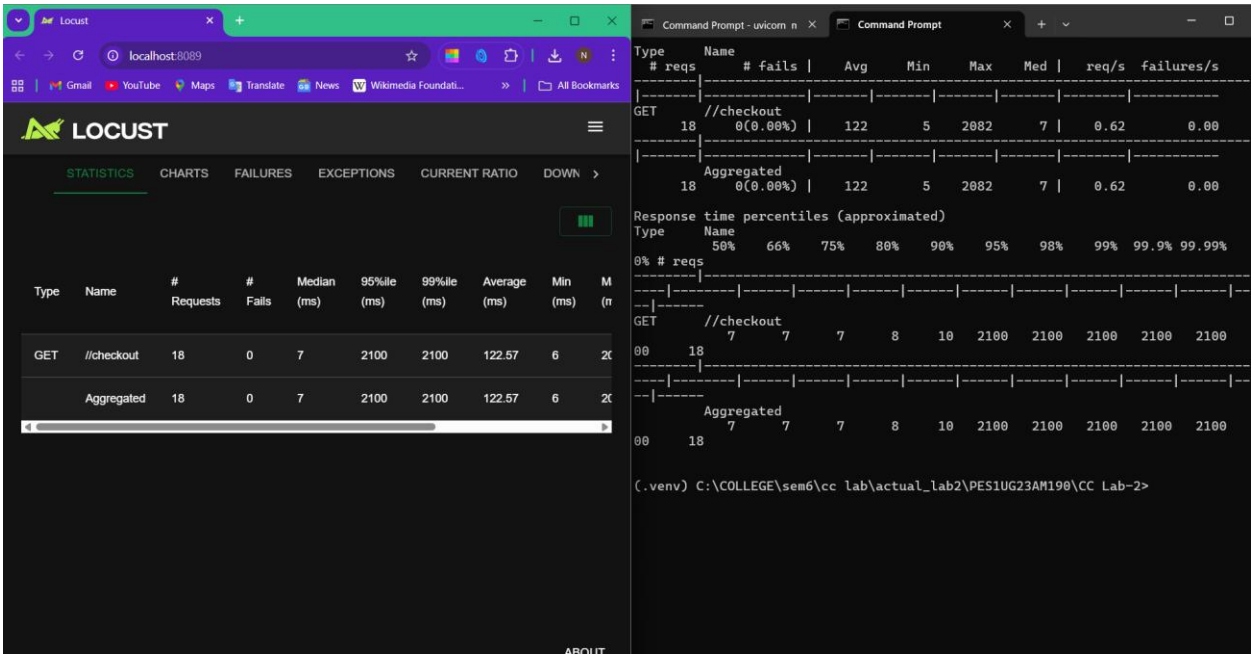
Average response time: 129.2ms



After Code optimisation



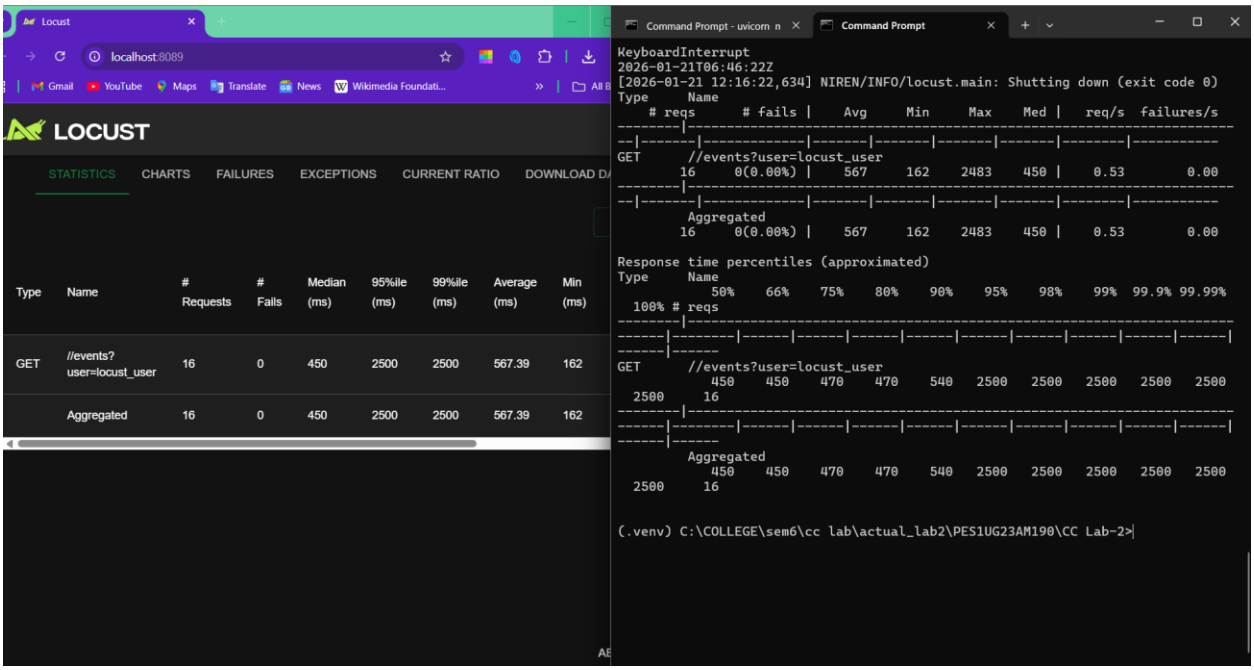
SS5: Reduced average response time: 122.57ms


Locust
localhost:8089
Gmail
YouTube
Maps
Translate
News
Wikimedia Foundati...
All Bookmarks
LOCUST
STATISTICS
CHARTS
FAILURES
EXCEPTIONS
CURRENT RATIO
DOWN
Type
Name
Requests
Fails
Median (ms)
95%ile (ms)
99%ile (ms)
Average (ms)
Min (ms)
GET
//checkout
18
0
7
2100
2100
122.57
6
Aggregated
18
0
7
2100
2100
122.57
6
ABOUT
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% 0% # reqs
GET //checkout 7 7 7 8 10 2100 2100 2100 2100 2100 00 18
Aggregated 7 7 7 8 10 2100 2100 2100 2100 2100 00 18
(.venv) C:\COLLEGE\sem6\cc lab\actual_lab2\PES1UG23AM190\CC Lab-2>

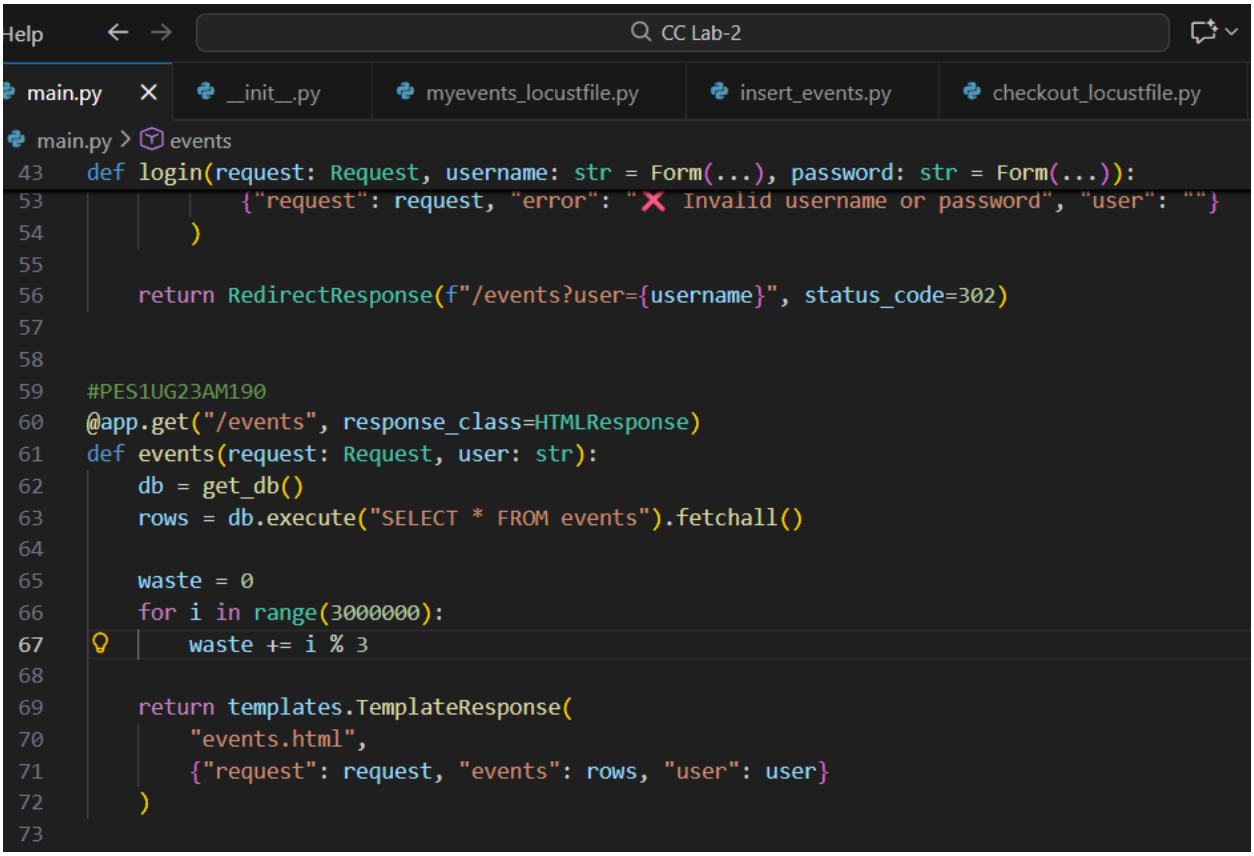
PART 7: Optimise events and my_events(DIY)

Route 1: /events

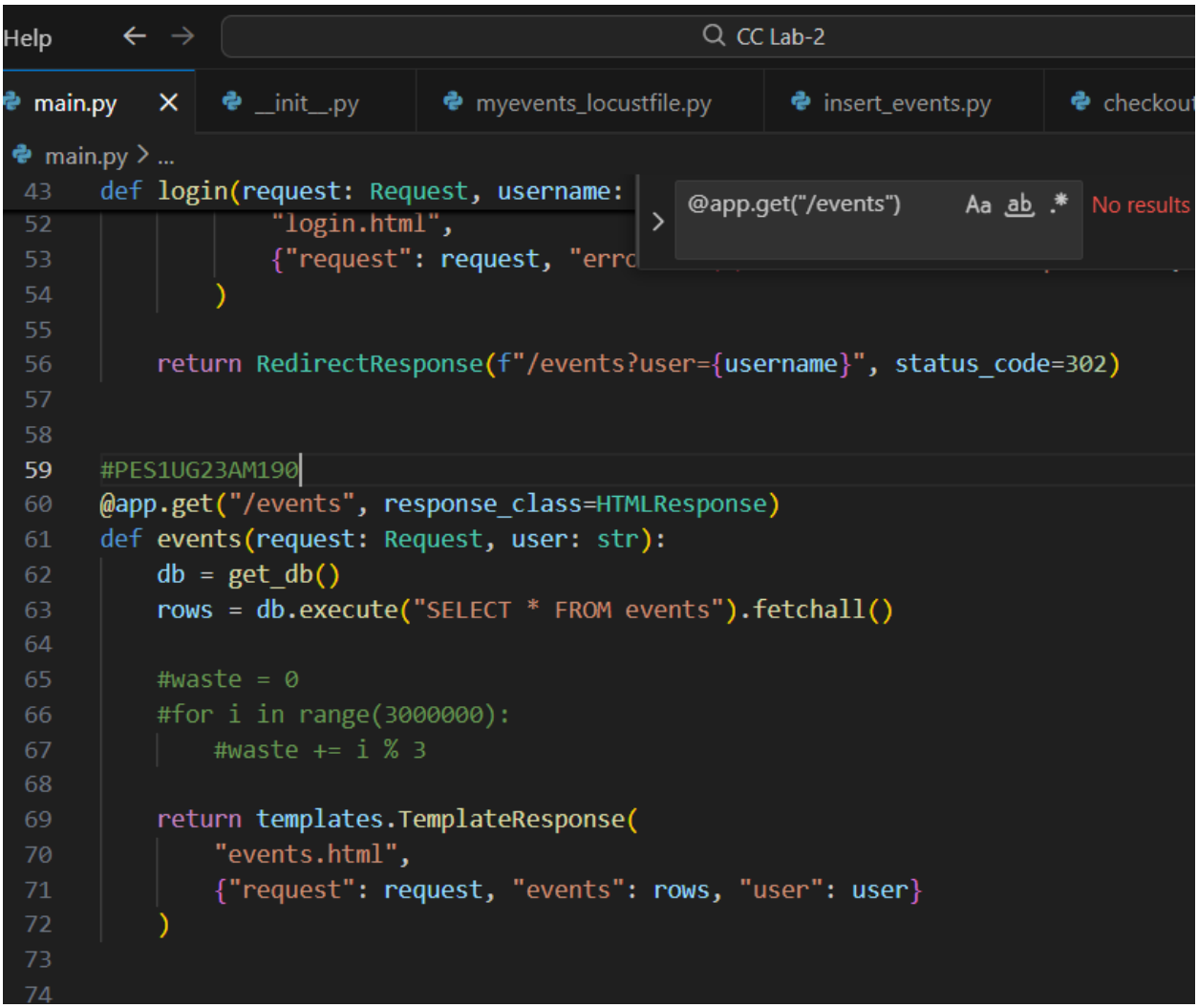
SS6: Screenshot BEFORE optimization(Avg response time:567.39ms)



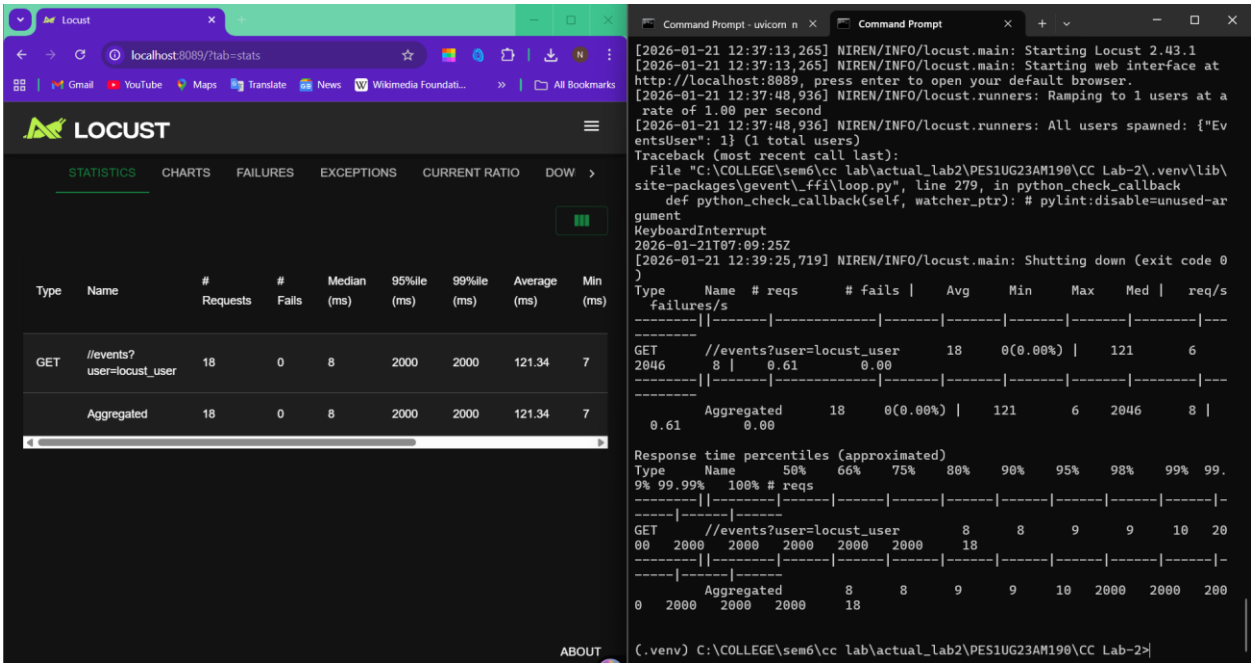
Code before optimization:



After optimizing code:



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?

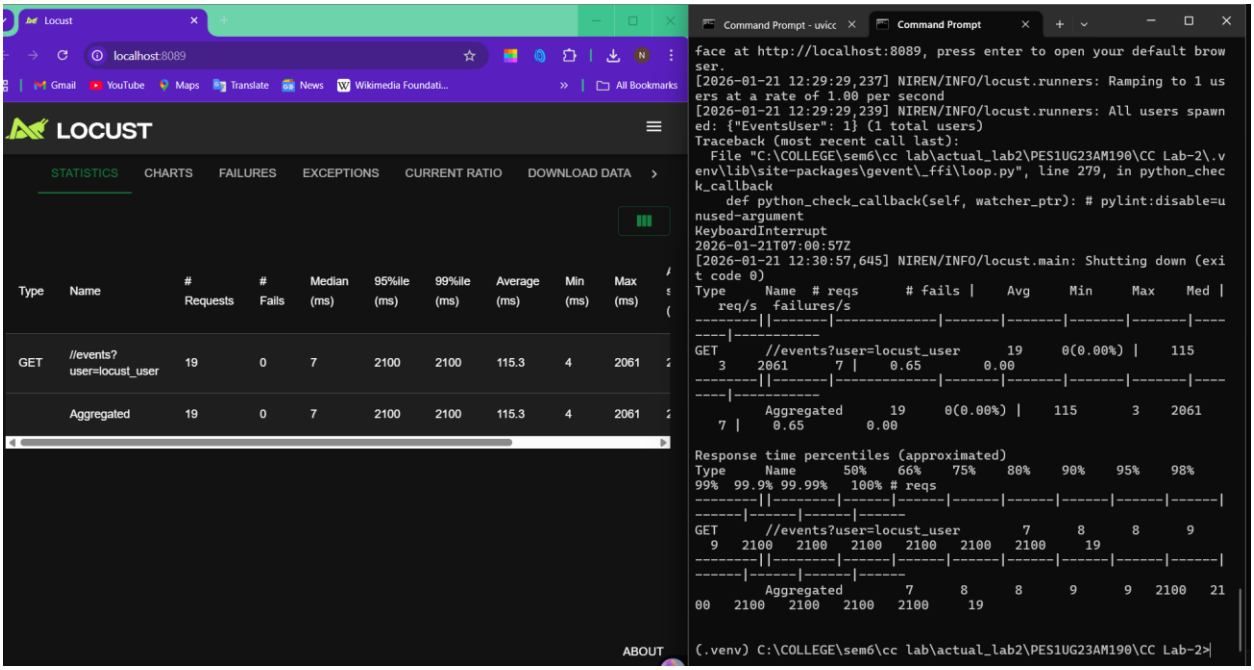
Ans: The redundant loop was removed so the route only performs the required database query.

3) Why did the performance improve?

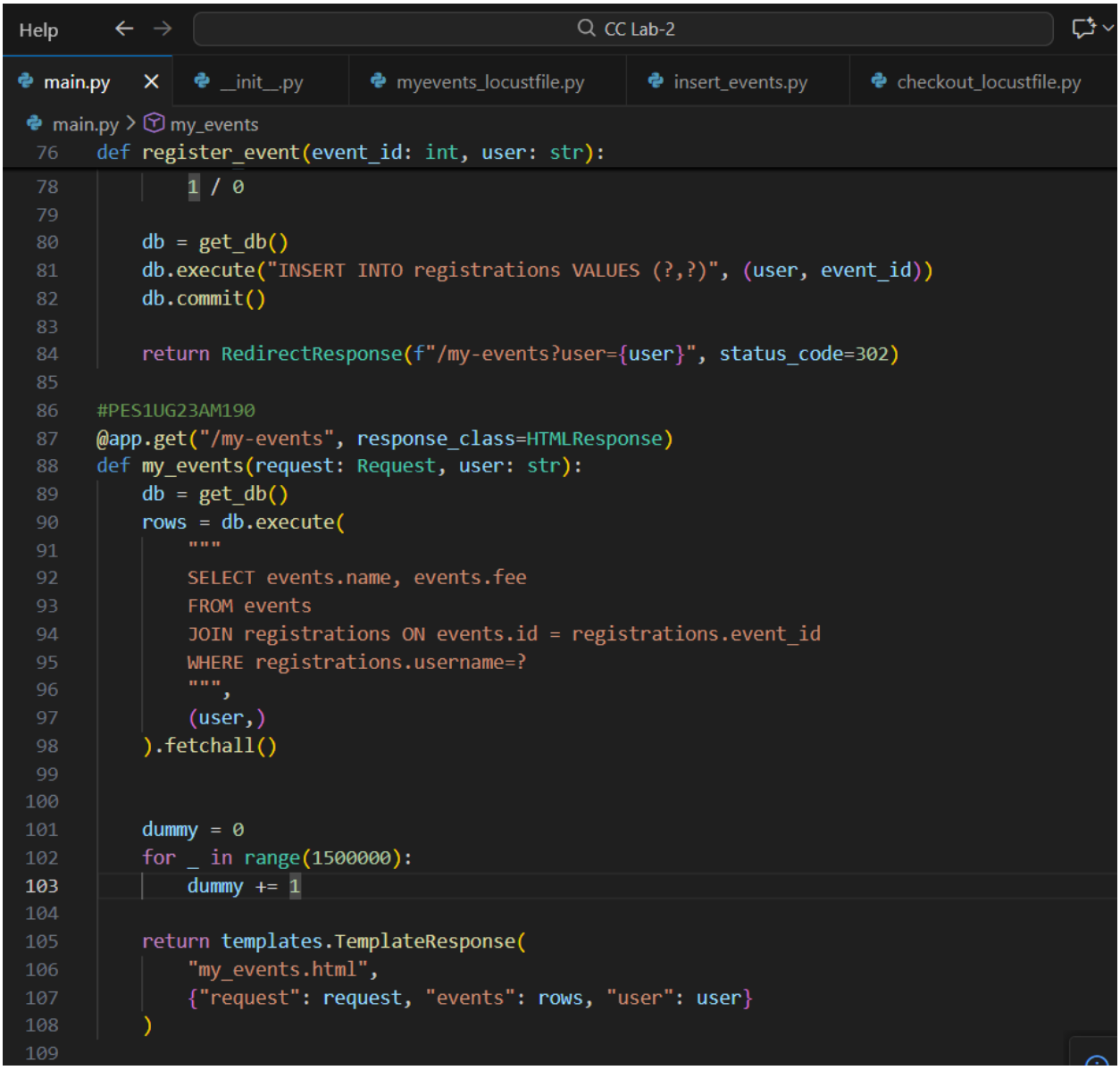
Ans: Removing the unnecessary computation reduced CPU usage and request processing time, allowing the server to handle more concurrent requests efficiently, thus reducing Average response time.

Route 2: /my-events

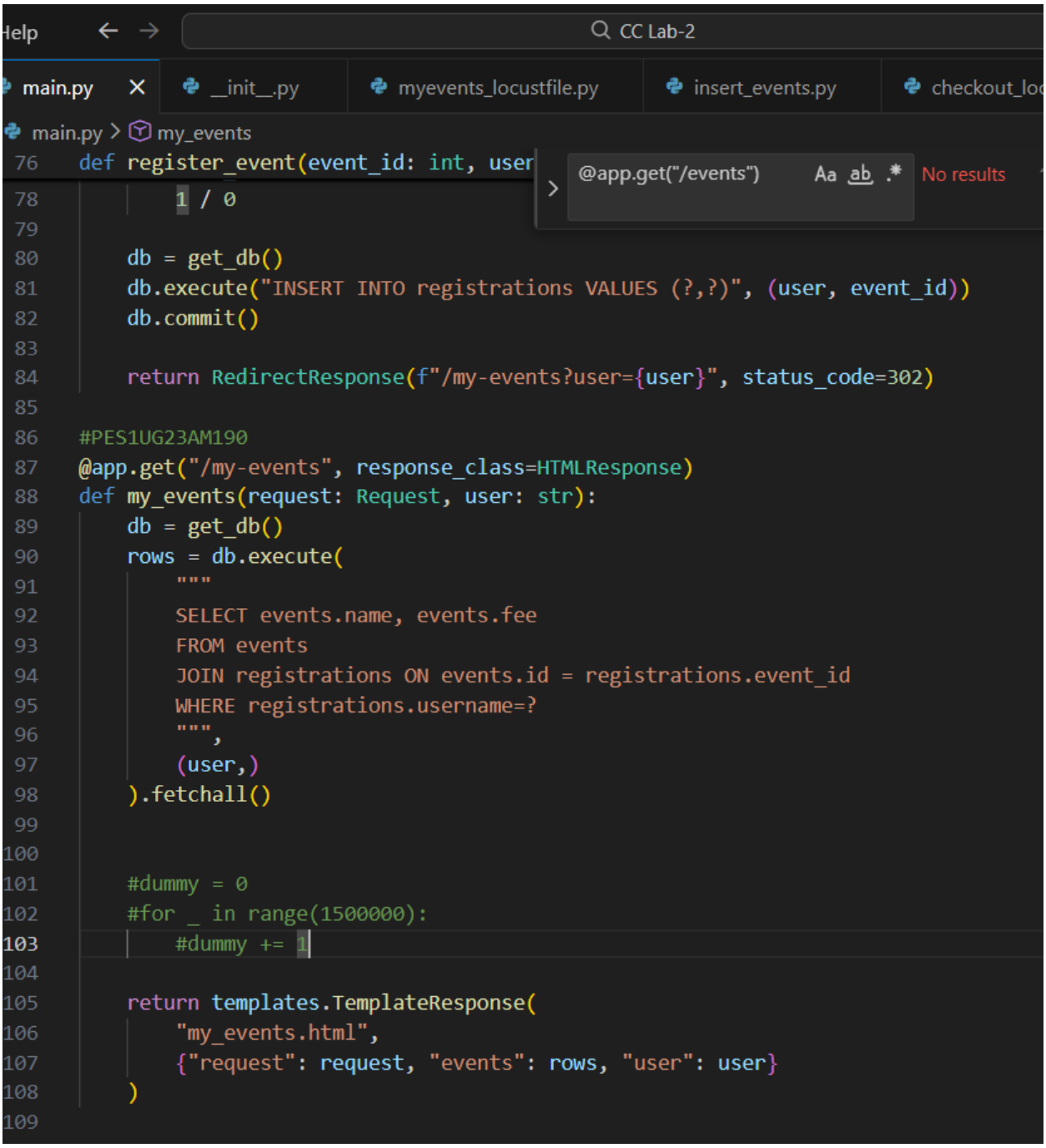
SS8: Screenshot BEFORE optimization (Avg response time:115.3 ms)



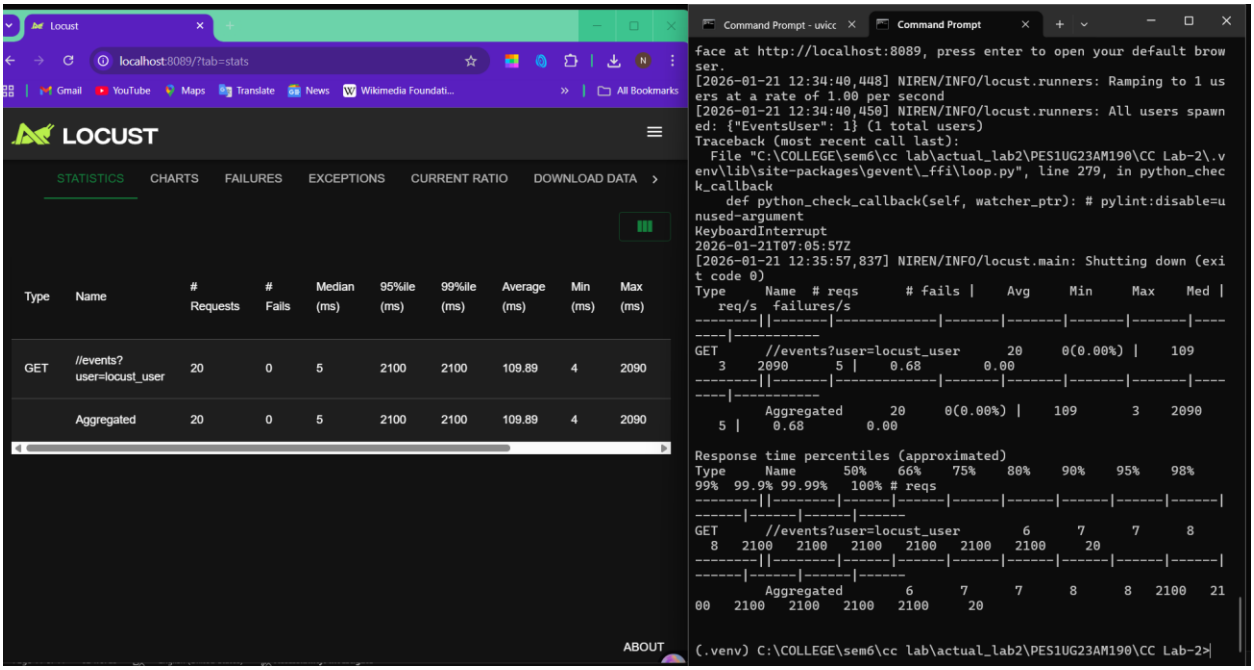
Code before optimization:



After optimizing code:



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.