



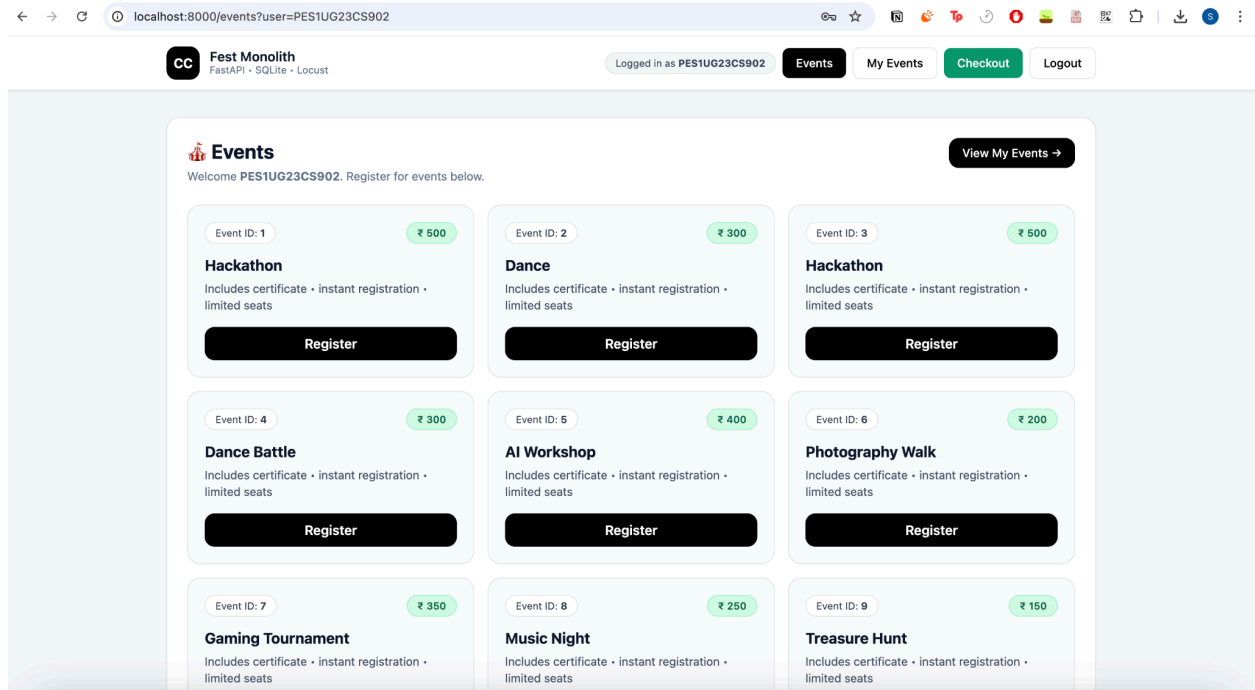
Department of Computer Science and Engineering

**UE23CS351B Cloud Computing**

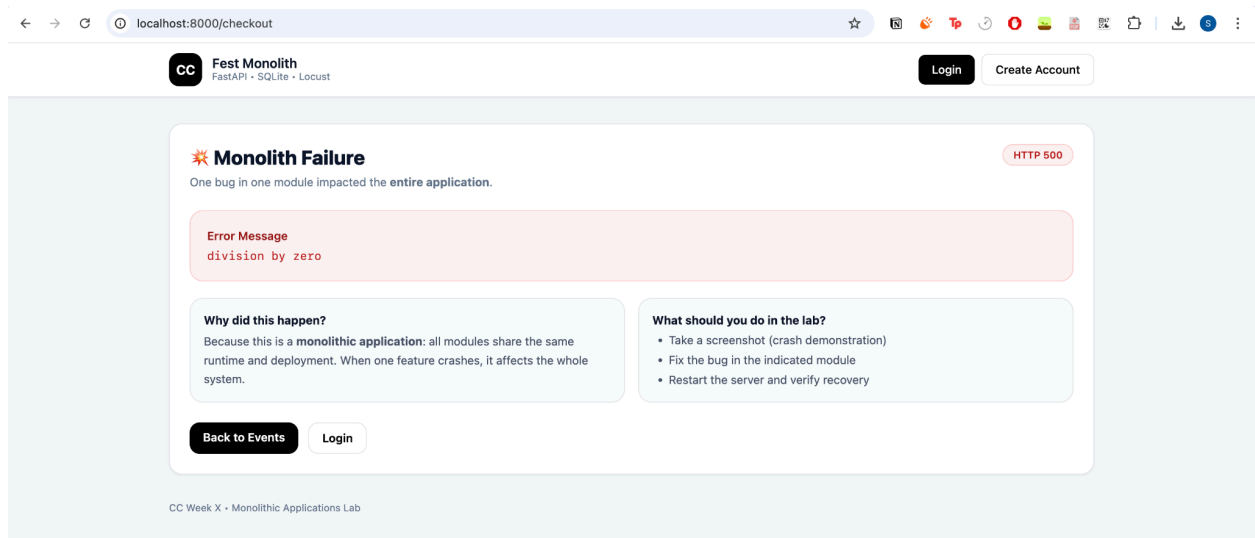
**Lab 2 Submission**

Name	Sudiksha Chindula
SRN	PES1UG23CS902
Section	L
Date	29 January 2026

## Screenshot SS1



## Screenshot SS2

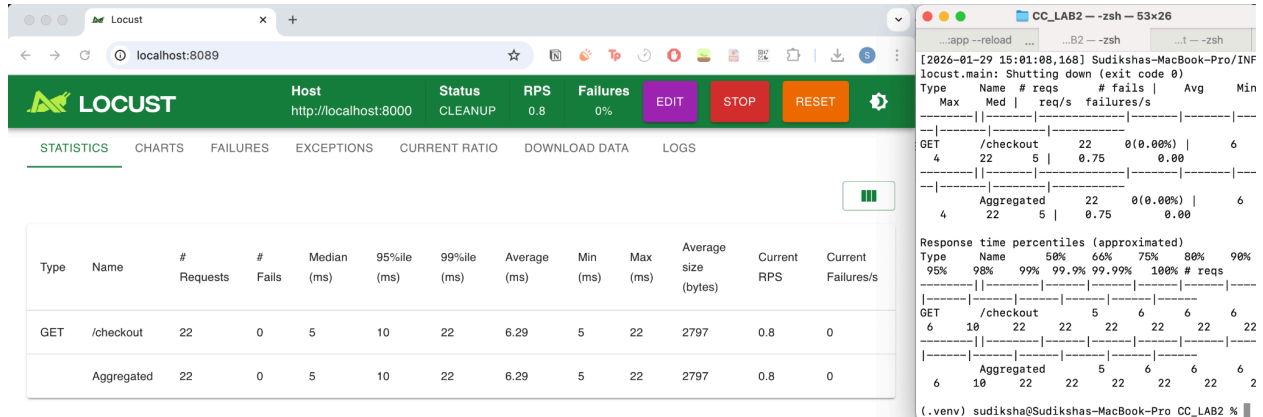


```
INFO: 127.0.0.1:49593 - "GET /events?user=PES1UG23CS902 HTTP/1.1" 200 OK
INFO: 127.0.0.1:49656 - "GET /checkout HTTP/1.1" 500 Internal Server Error
ERROR: Exception in ASGI application
Traceback (most recent call last):
```

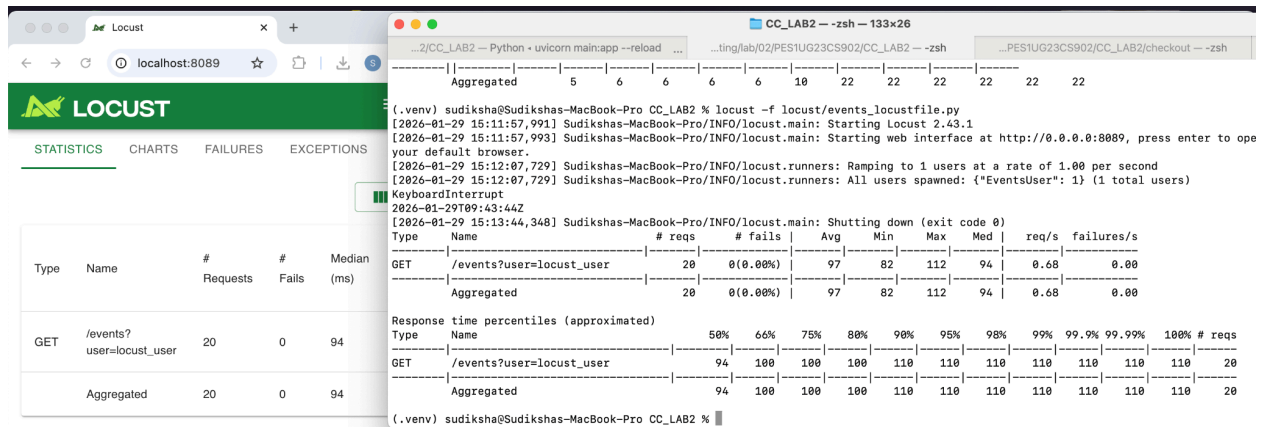
The screenshot displays the Locust web interface in a browser window at localhost:8089. The interface shows a summary of the load test: Host is http://localhost:8000, Status is CLEANUP, RPS is 0.6, and Failures are 0%. Below this, there are tabs for STATISTICS, CHARTS, FAILURES, EXCEPTIONS, CURRENT RATIO, and DOWNLOAD DATA. The STATISTICS tab is active, showing a table with columns: Type, Name, # Requests, # Fails, Median (ms), 95%ile (ms), 99%ile (ms), Average (ms), Min (ms), and Max (ms). The table contains two rows: GET /checkout and Aggregated. The GET /checkout row shows 19 requests, 0 fails, a median of 6ms, and an average of 5.82ms. The Aggregated row shows 19 requests, 0 fails, a median of 6ms, and an average of 5.82ms. To the right of the browser window, a terminal window shows the command output for the Locust load test. The terminal output includes the command 'n -u uvicorn main:app --reload ... CC\_LAB2 -- -zsh -- 89x35', the command 'n -u uvicorn main:app --reload ... CC\_LAB2 -- -zsh', and the command 'n -u uvicorn main:app --reload ... CC\_LAB2 -- -zsh'. The output shows the Locust interface starting, the test running, and the test results.

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)
GET	/checkout	19	0	6	9	9	5.82	3	9
	Aggregated	19	0	6	9	9	5.82	3	9

## Screenshot SS5

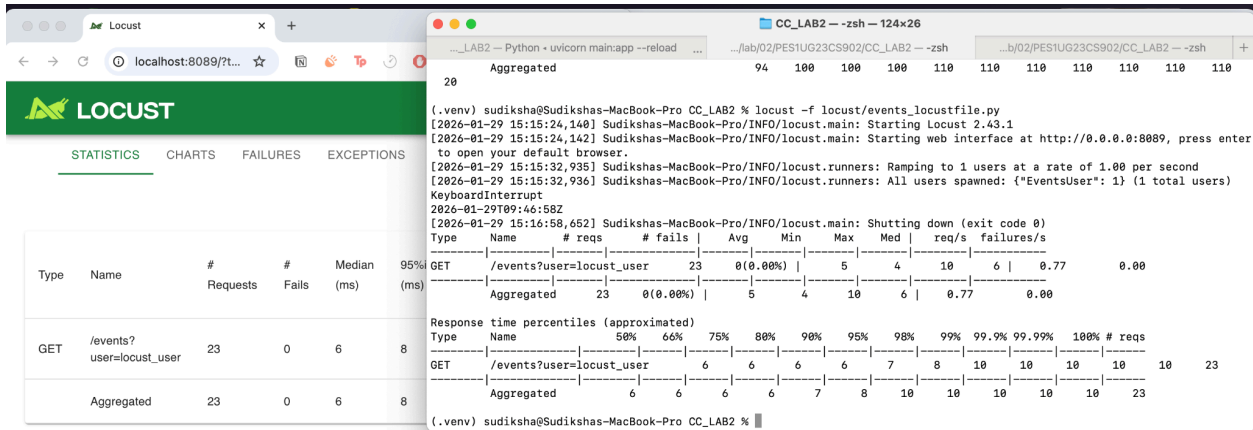


## Screenshot SS6

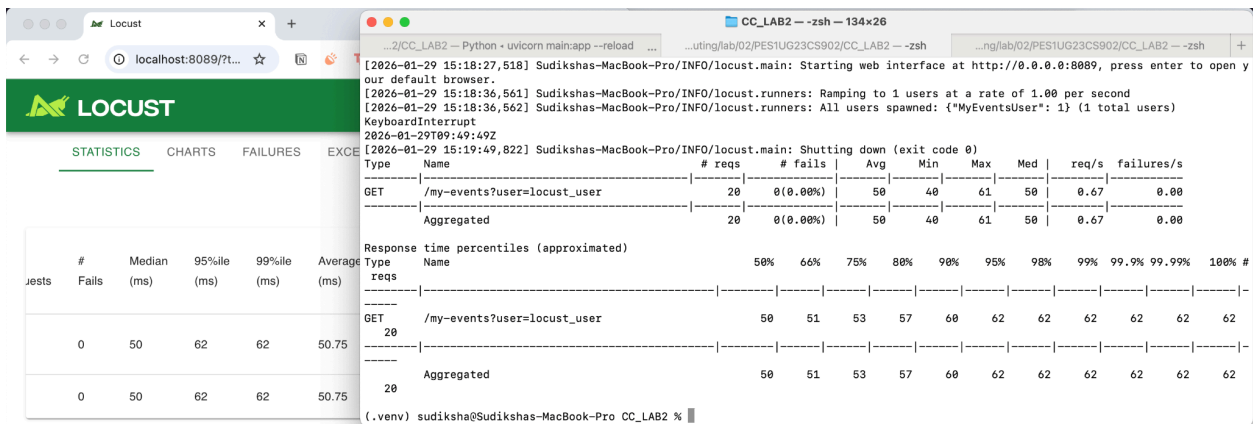


Optimisation: removed for loop that updated variable 'waste' over several iterations

## Screenshot SS7

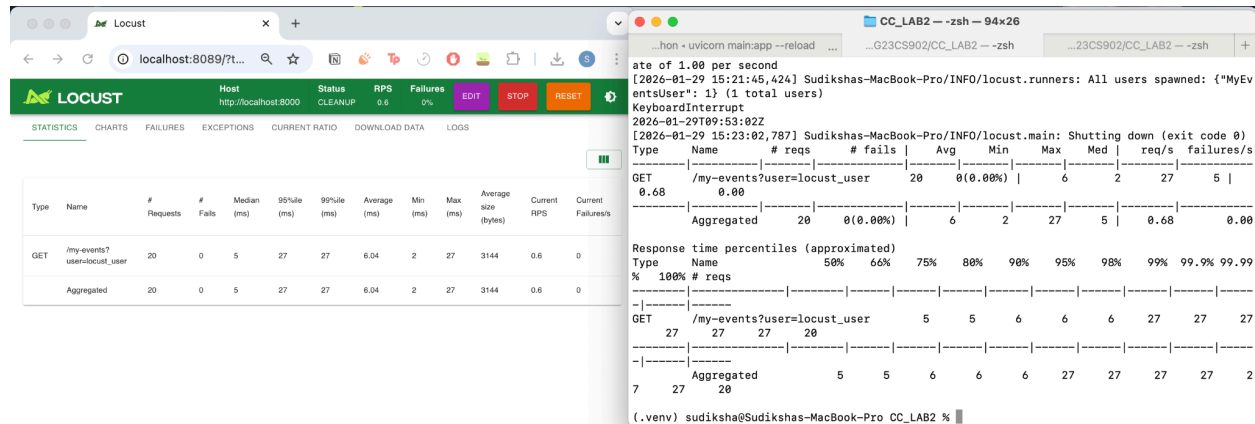


## Screenshot SS8



Optimised code: removed for loop that updated variable 'dummy'

## Screenshot SS9



## Explanation of optimisations:

### Route 1: /events

**Bottleneck:** The function contained a CPU-intensive for loop iterating 3,000,000 times to calculate a waste variable, which served no functional purpose.

**Change:** The unnecessary loop and the waste variable calculation were completely removed.

**Improvement:** Removing the loop eliminated the artificial CPU processing delay, allowing the server to return the template response immediately after fetching data from the database. Reduced from an Average of 97 to 23

### Route 2: /my-events

**Bottleneck:** The function included a for loop iterating 1,500,000 times to increment a dummy variable, creating an artificial delay.

**Change:** The dummy variable loop was identified as dead code and removed.

**Why it improved:** By removing the blocking loop, the request handling time was reduced to just the database query and template rendering time, significantly lowering latency from an average of 50ms to 6ms