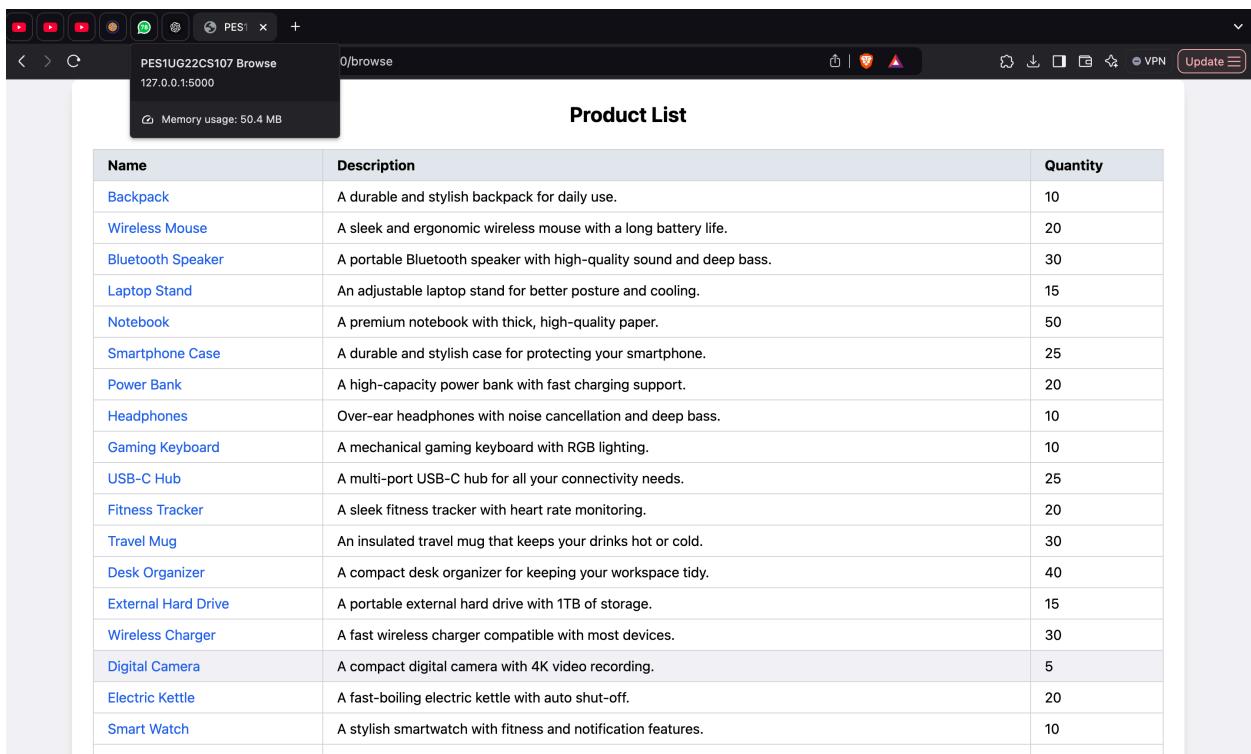


UE22CS351B CLOUD COMPUTING LABORATORY 3

NAME : ARNAV SATISH
SRN : PES1UG22CS107

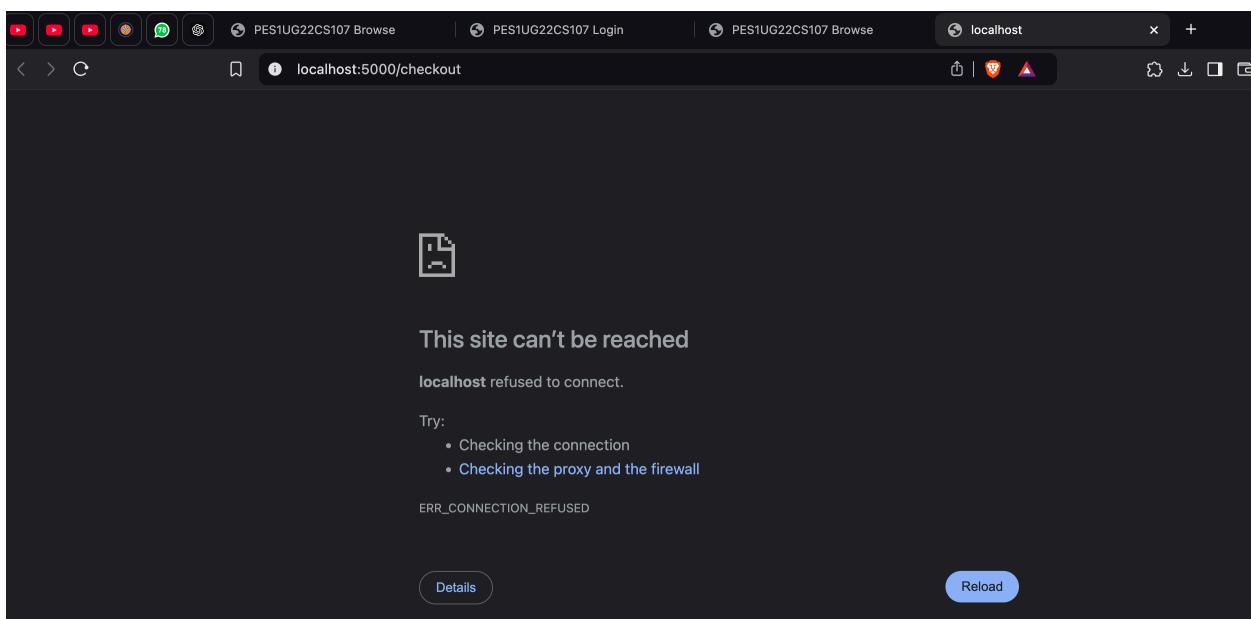
SS1:



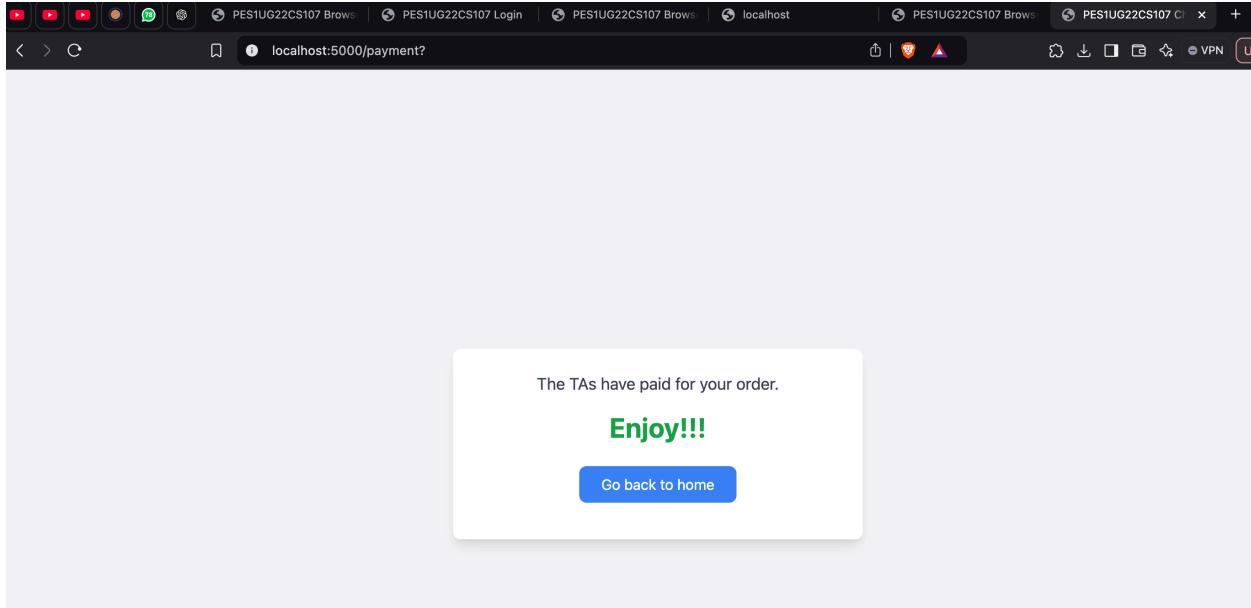
A screenshot of a web browser window titled "PES1UG22CS107 Browse 127.0.0.1:5000". The page displays a "Product List" table with columns for Name, Description, and Quantity. The table contains 20 rows of product information.

Name	Description	Quantity
Backpack	A durable and stylish backpack for daily use.	10
Wireless Mouse	A sleek and ergonomic wireless mouse with a long battery life.	20
Bluetooth Speaker	A portable Bluetooth speaker with high-quality sound and deep bass.	30
Laptop Stand	An adjustable laptop stand for better posture and cooling.	15
Notebook	A premium notebook with thick, high-quality paper.	50
Smartphone Case	A durable and stylish case for protecting your smartphone.	25
Power Bank	A high-capacity power bank with fast charging support.	20
Headphones	Over-ear headphones with noise cancellation and deep bass.	10
Gaming Keyboard	A mechanical gaming keyboard with RGB lighting.	10
USB-C Hub	A multi-port USB-C hub for all your connectivity needs.	25
Fitness Tracker	A sleek fitness tracker with heart rate monitoring.	20
Travel Mug	An insulated travel mug that keeps your drinks hot or cold.	30
Desk Organizer	A compact desk organizer for keeping your workspace tidy.	40
External Hard Drive	A portable external hard drive with 1TB of storage.	15
Wireless Charger	A fast wireless charger compatible with most devices.	30
Digital Camera	A compact digital camera with 4K video recording.	5
Electric Kettle	A fast-boiling electric kettle with auto shut-off.	20
Smart Watch	A stylish smartwatch with fitness and notification features.	10

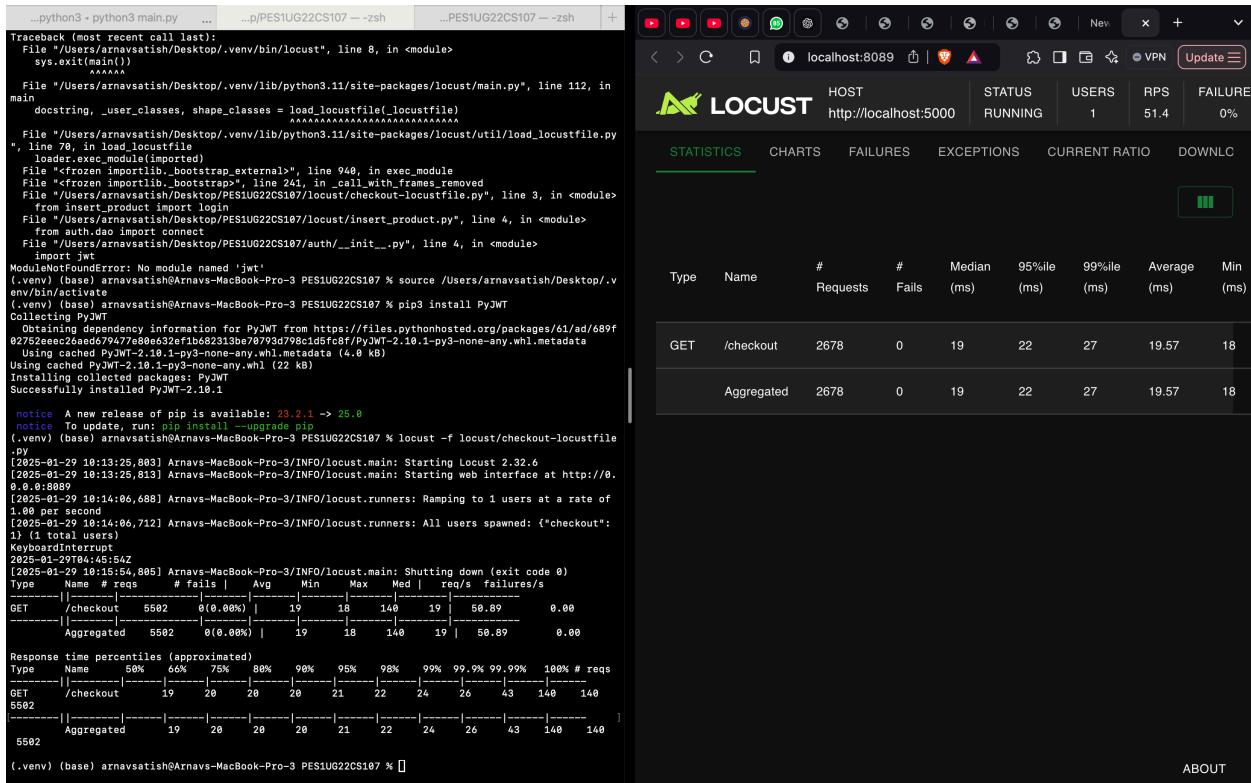
SS2:



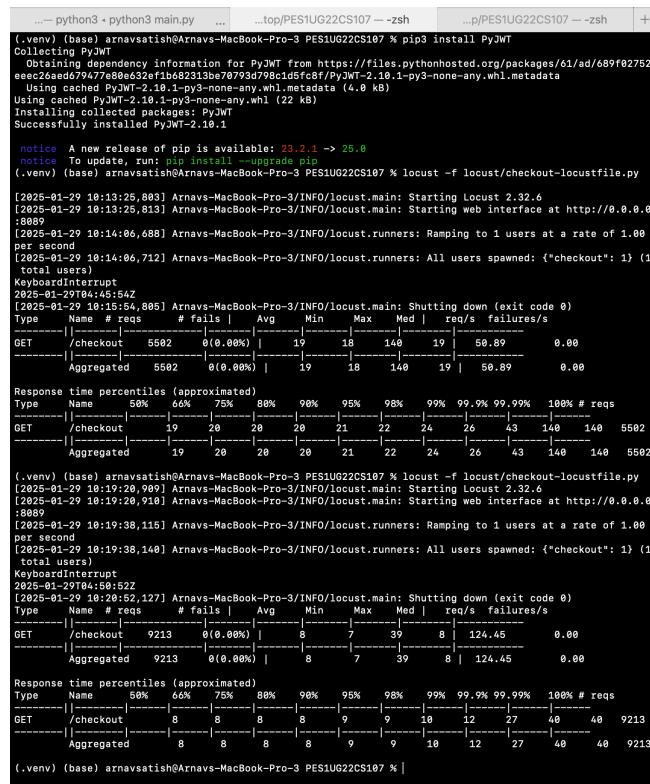
SS3:



SS4:



SS5:



The terminal output shows the execution of a Locust test. The command used was `python3 -m locust --host http://localhost:5000 --users 1 --spawn-rate 1 --run-time 10s`. The output includes logs from both the client and server sides, showing response time percentiles and a summary table.

Type	Name	# reqs	# fails	Avg	Min	Max	Med	req/s	failures/s
GET	/checkout	5502	0 (0.0%)	19	18	140	19	50.89	0.00
	Aggregated	5502	0 (0.0%)	19	18	140	19	50.89	0.00

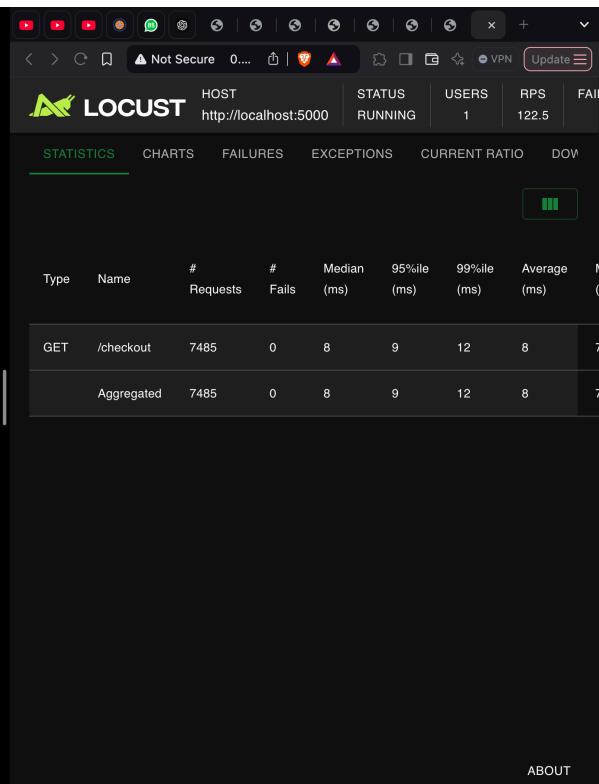
Response time percentiles (approximated)

Type	Name	50%	66%	75%	80%	90%	95%	98%	99%	99.9%	99.99%	100%	# reqs
GET	/checkout	19	20	20	20	21	22	24	26	43	140	140	5502
	Aggregated	19	20	20	20	21	22	24	26	43	140	140	5502

Response time percentiles (approximated)

Type	Name	50%	66%	75%	80%	90%	95%	98%	99%	99.9%	99.99%	100%	# reqs
GET	/checkout	8	8	8	8	9	9	10	12	27	40	40	9213
	Aggregated	8	8	8	8	9	9	10	12	27	40	40	9213

(.venv) (base) arnavs@arnavsmacbook-Pro-3 PES1UG22CS107 %



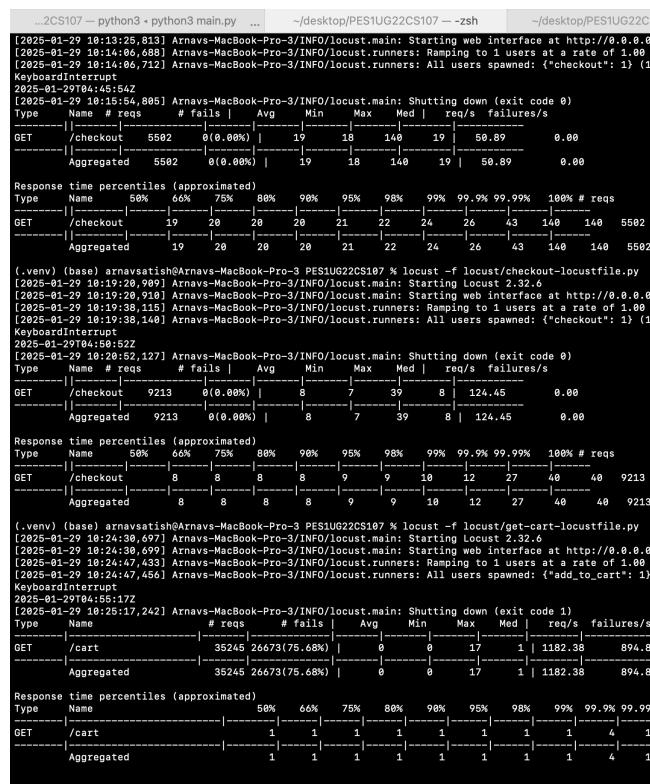
LOCUST HOST http://localhost:5000 STATUS RUNNING USERS 1 RPS 122.5 FAILS 0

STATISTICS

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)
GET	/checkout	7485	0	8	9	12	8
	Aggregated	7485	0	8	9	12	8

ABOUT

SS6:



The terminal output shows the execution of a Locust test. The command used was `python3 -m locust --host http://localhost:5000 --users 1 --spawn-rate 1 --run-time 10s`. The output includes logs from both the client and server sides, showing response time percentiles and a summary table.

Type	Name	# reqs	# fails	Avg	Min	Max	Med	req/s	failures/s
GET	/checkout	5502	0 (0.0%)	19	18	140	19	50.89	0.00
	Aggregated	5502	0 (0.0%)	19	18	140	19	50.89	0.00

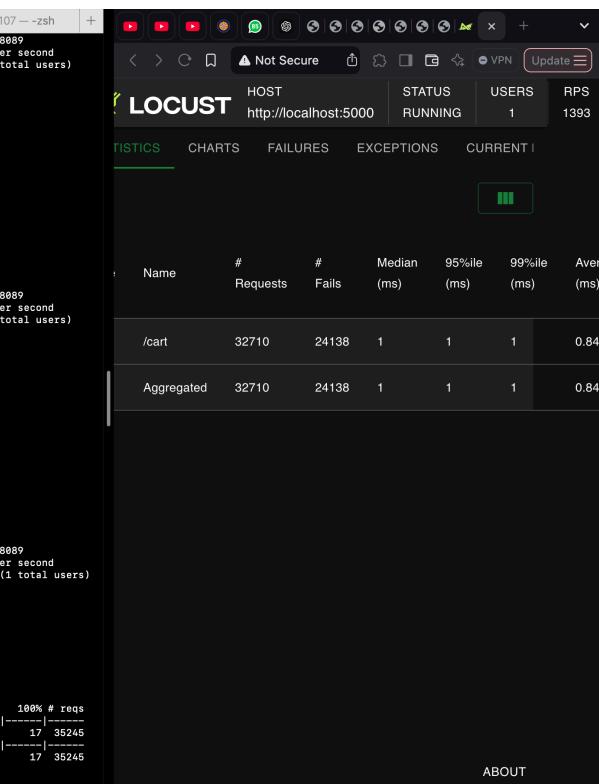
Response time percentiles (approximated)

Type	Name	50%	66%	75%	80%	90%	95%	98%	99%	99.9%	99.99%	100%	# reqs
GET	/checkout	19	20	20	20	21	22	24	26	43	140	140	5502
	Aggregated	19	20	20	20	21	22	24	26	43	140	140	5502

Response time percentiles (approximated)

Type	Name	50%	66%	75%	80%	90%	95%	98%	99%	99.9%	99.99%	100%	# reqs
GET	/checkout	8	8	8	8	9	9	10	12	27	40	40	9213
	Aggregated	8	8	8	8	9	9	10	12	27	40	40	9213

(.venv) (base) arnavs@arnavsmacbook-Pro-3 PES1UG22CS107 %



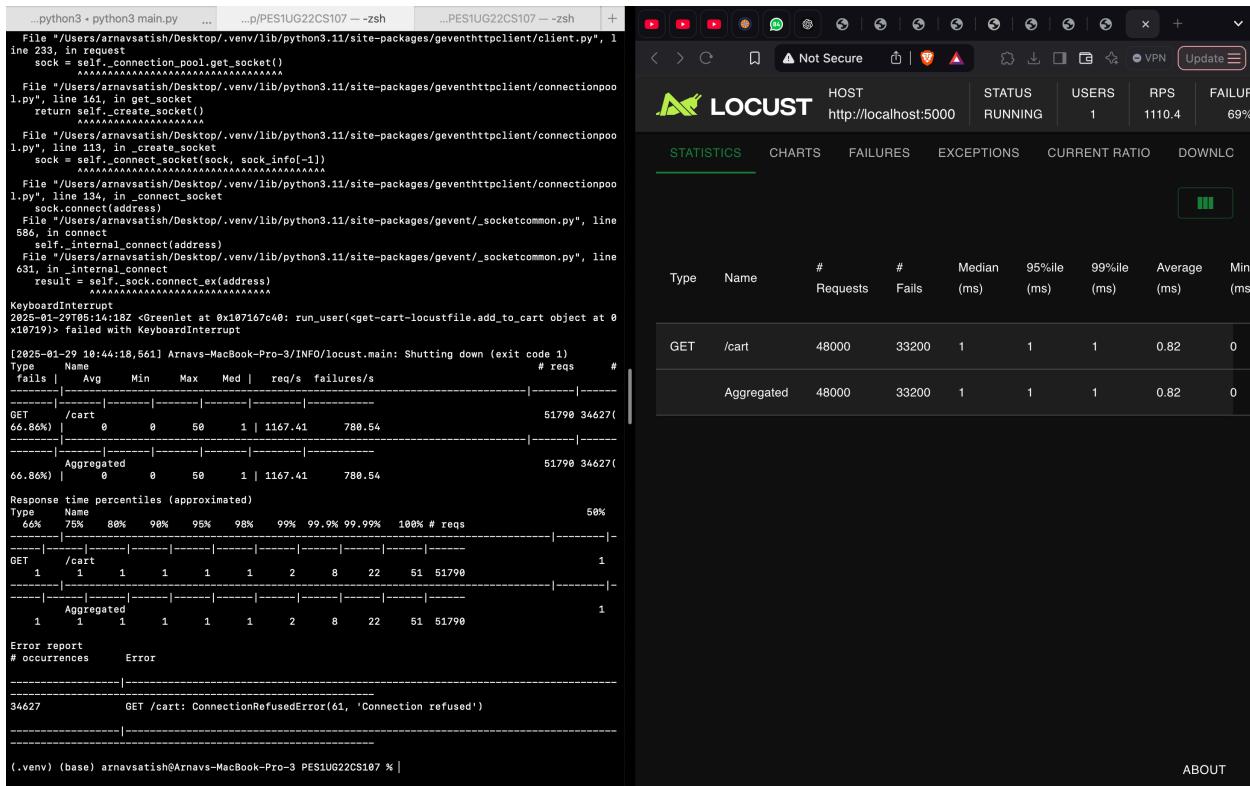
LOCUST HOST http://localhost:5000 STATUS RUNNING USERS 1 RPS 1393 FAILS 0

STATISTICS

Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	
/cart	32710	24138	1	1	1	0.84	
	Aggregated	32710	24138	1	1	1	0.84

ABOUT

SS7:



```
def get_cart(username: str) -> list:
    conn = connect('carts.db')
    cursor = conn.cursor()
    if cursor:
        cursor.execute('SELECT * FROM carts WHERE username = ?', (username,))
    else:
        return []

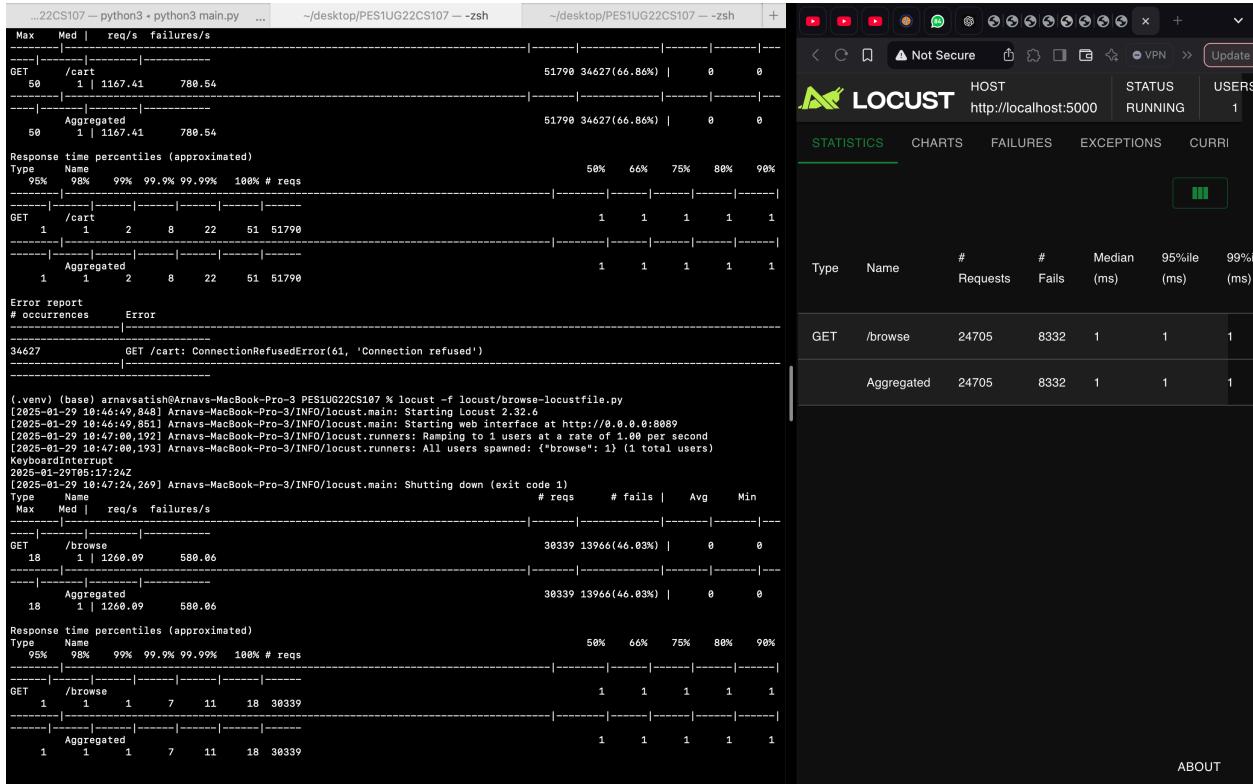
    cart = cursor.fetchall()
    temp_cart = []
    for row in cart:
        temp_cart.append(row)

    final_cart = []
    for item in temp_cart:
        final_cart.append(item)
    final_cart=cursor.fetchall()

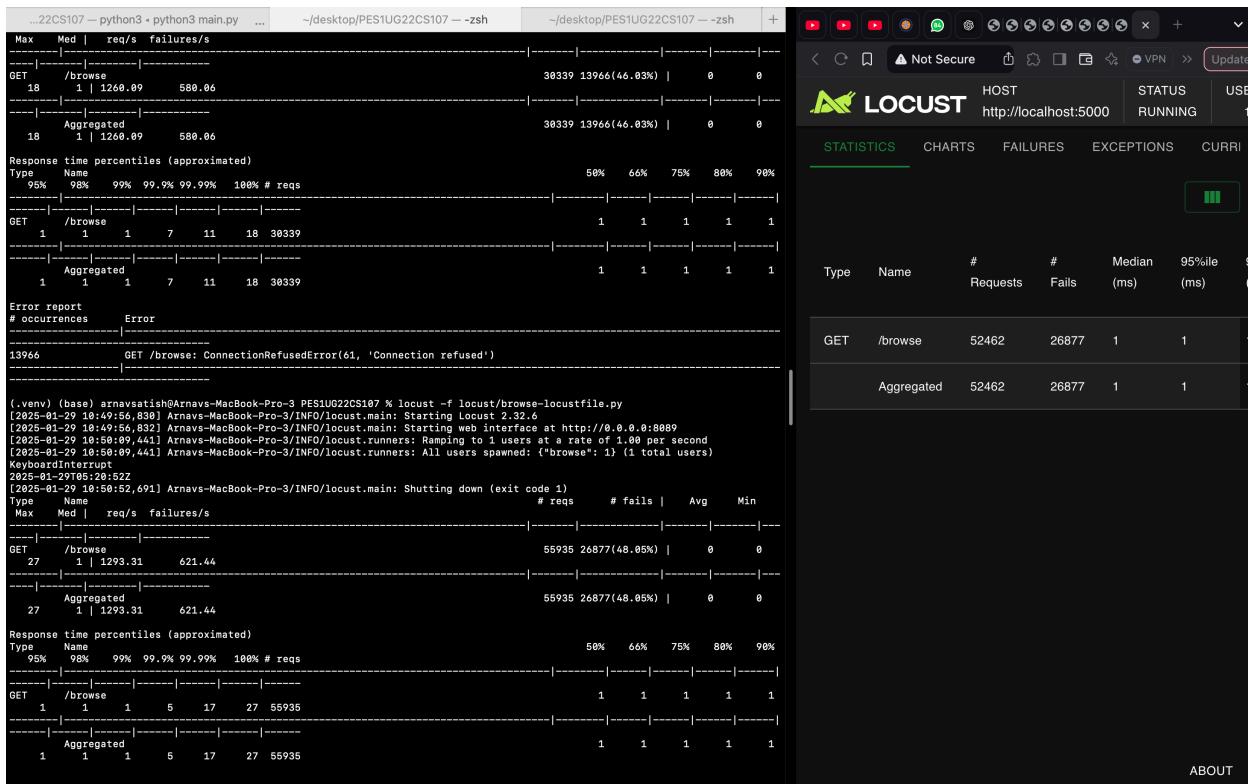
    cursor.close()
    conn.close()
    return final_cart
```

The provided code has unwanted for loop. This was optimized by using line of code
“final_cart=cursor.fetchall()”

SS8:



SS9:



```

def list_products():
    conn = connect('products.db')
    cursor = conn.cursor()

    cursor.execute('SELECT * FROM products')

    '''products = []
    rows = cursor.fetchall()

    for i in range(len(rows)):
        temp = rows[i]
        products.append(temp)'''
    products=cursor.fetchall()

    cursor.close()
    conn.close()
    if len(products) > 0:
        products.sort(key=lambda x: 0)
    return products

```

The provided code has unwanted for loop. This was optimized by using line of code “products=cursor.fetchall()”