Locust Load Testing Report

1. Overview

This report summarizes the results of a Locust load test conducted on the Fashion MNIST Predictor web application. The purpose was to measure how the system performs under simulated user traffic and identify potential bottlenecks.

2. Test Setup

Tool: Locust 2.37.14

Host: http://127.0.0.1:8000

Test Interface: http://localhost:8089

Test Scenarios:

- GET request to the homepage ('/')

- POST request to the prediction endpoint ('/predict')

The test gradually increased the number of users to observe system performance.

3. Results

Total Requests: 847

Failure Rate: 100% (all requests failed)

Median Response Time: ~5.3 seconds

95th Percentile: ~9.6 seconds

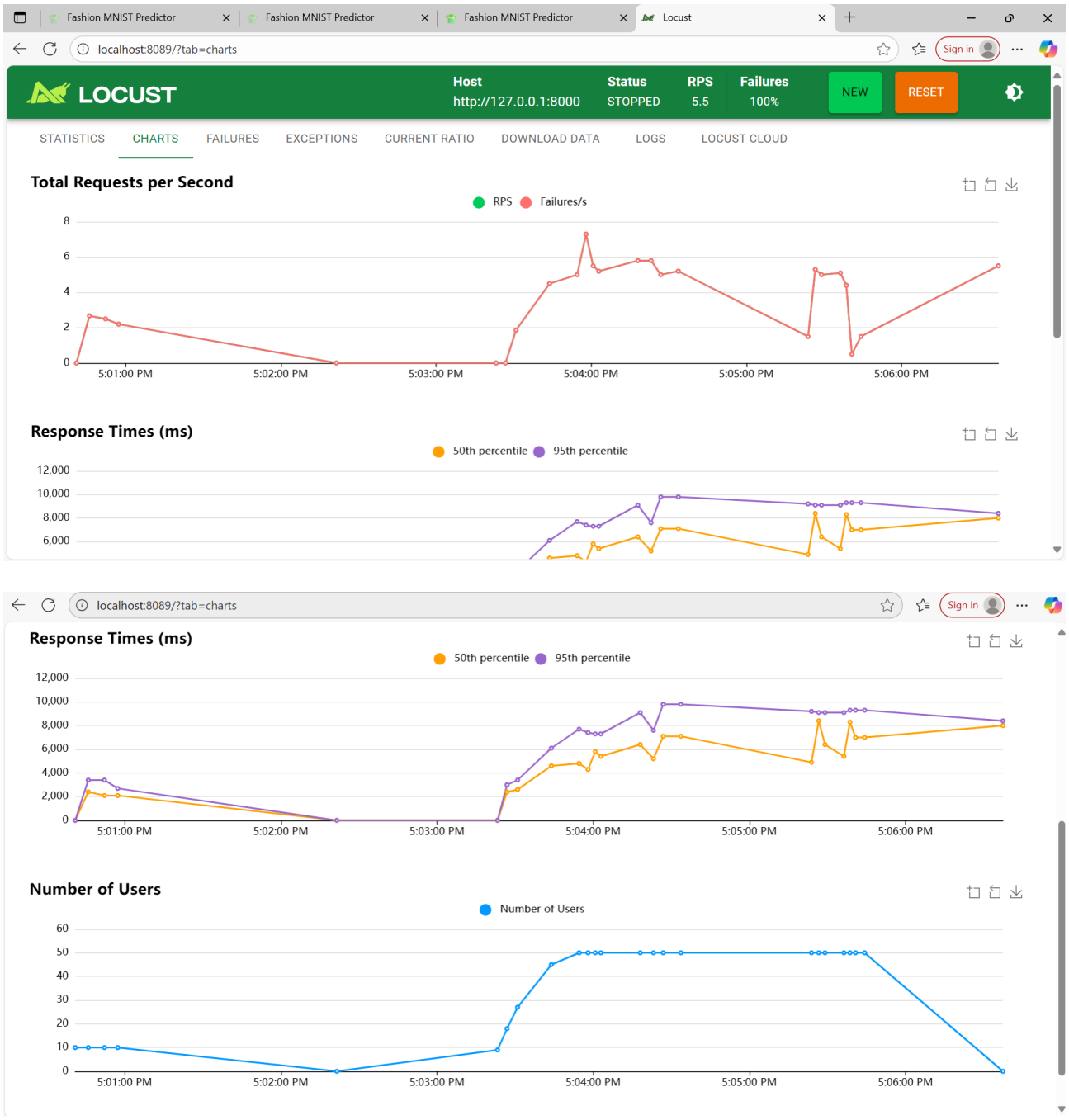
99th Percentile: ~11 seconds

Requests Per Second (RPS): ~5.5 at peak load

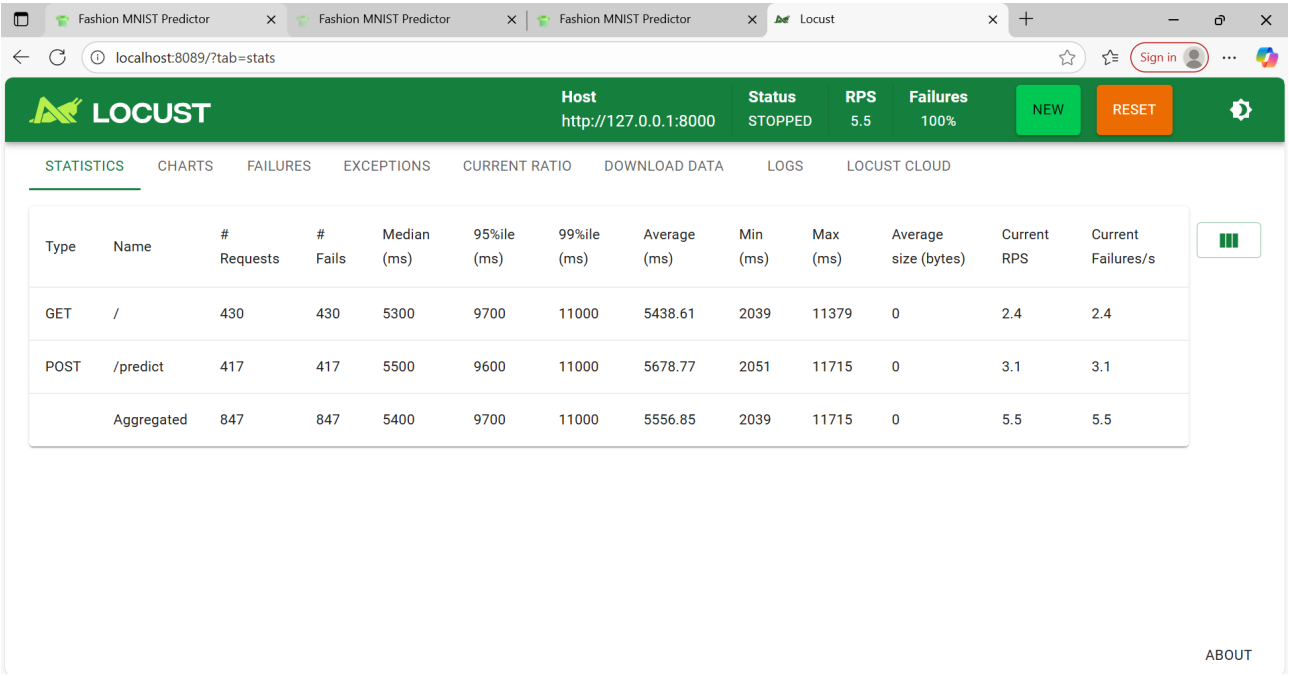
4. Charts and Visuals

The following charts show the request rates, response times, and error statistics during the test.

Locust Load Testing Report



Locust Load Testing Report



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	/	430	430	5300	9700	11000	5438.61	2039	11379	0	2.4	2.4
POST	/predict	417	417	5500	9600	11000	5678.77	2051	11715	0	3.1	3.1
	Aggregated	847	847	5400	9700	11000	5556.85	2039	11715	0	5.5	5.5

5. Analysis

The application struggled to handle the simulated load. High response times and a 100% failure rate suggest server or application-level performance issues. Possible reasons include:

- Model loading for each request instead of once at startup
- Long processing times for predictions
- Lack of concurrency handling

6. Recommendations

- Load the prediction model once at startup
- Use asynchronous processing for requests
- Cache results for repeated predictions
- Deploy using a production server (e.g., Gunicorn or Uvicorn with workers)
- Scale infrastructure resources

7. Conclusion

The Locust load test revealed significant performance challenges. Implementing the above optimizations should improve response times, reduce failures, and enhance the system's ability to handle higher traffic.