Performance Optimization Report

Python API – MySQL – Redis – Docker – k6

1. Project Objective

The objective of this project is to identify backend performance bottlenecks and demonstrate measurable improvements in response time, throughput, and scalability using industry-standard DevOps and backend optimization practices.

2. Technology Stack

Backend: Python (Flask)
Database: MySQL 8
Cache: Redis
DevOps: Docker, Docker Compose
Load Testing: k6

3. System Architecture

The following architecture represents a typical high-performance backend system. Caching is introduced between the application and database to reduce load and improve response times.

Client
↓ HTTP
Flask API (Docker Container)
↓ Cache Lookup
Redis (In-Memory Cache)
↓ Cache Miss
MySQL Database

4. Load Testing Scenario

Load testing was performed using k6 with 50 concurrent virtual users for a duration of two minutes. The same infrastructure and configuration were used for both baseline and optimized tests to ensure fairness.

5. Performance Metrics – Before Optimization

Average response time was approximately 363 ms, with p95 latency around 434 ms. Throughput was limited to about 36 requests per second. The database was identified as the primary bottleneck.

6. Performance Metrics – After Optimization

After introducing Redis caching, the average response time dropped to approximately 8.9 ms, with p95 latency under 9 ms. Throughput increased to around 49 requests per second, representing a significant performance improvement without additional infrastructure cost.

Metric	Before	After	Improvement
Average Response Time	363 ms	8.9 ms	~40x faster
p95 Latency	434 ms	8.3 ms	Massive reduction
Throughput	36 req/s	49 req/s	~36% increase
Error Rate	0%	0%	No regression

7. Conclusion

This project demonstrates how targeted architectural improvements, particularly caching, can drastically improve backend performance. The results are production-realistic, measurable, and directly applicable to ERP systems, SaaS platforms, and business-critical APIs.