PERFORMANCE MAJOR PROJECTS

Web Application Performance Testing

Problem Statement: Web application performance was sluggish with frontend bottlenecks and inconsistent load handling.

Tech Stack: JMeter, Google PageSpeed Insights, Lighthouse

Interface: Web (Frontend performance testing)

Role & Contributions:

- Simulated HTTPS load using JMeter Distributed (Controller-Worker setup).
- Stress-tested application performance across load variations.
- Utilized JMeter listeners and custom plugins to visualize throughput, latency, and errors.
- Identified render-blocking issues and optimized static assets.

- Provided actionable insights that helped reduce frontend load time by 27% and enhanced user experience.
- Integrated performance checks for critical UI journeys pre-release.

Backend Service Performance Testing

Problem Statement: Backend APIs degraded under moderate traffic, with no prior load or stress testing in place.

Tech: Locust, K6

Type: API Load & Stress Testing

- Role & Contributions:
- Developed Locust-based scripts for TPS and concurrency benchmarking.
- Evaluated service health under load for infrastructure scaling validation.
- Provided detailed latency, error rate, and throughput analysis report.
- Dockerized the Locust script for execution via Docker Compose and distributed mode to simulate target load.

- Built stress-test baselines to uncover backend bottlenecks under high concurrency.
- Cut performance incidents through early scalability validation and proactive planning.

Database Performance Testing

Problem Statement: Identify the most suitable database for a new backend service by evaluating multiple implementations based on service requirements and performance under load.

Tech: Locust, Custom CRUD API, Multiple Database Engines

Interface: API, different databases and configurations.

Role & Contributions:

- Simulated DB interactions using custom APIs to mimic real-world load conditions.
- Designed tailored CRUD operations to benchmark latency, throughput.
- Compared multiple database technologies across varied configurations.
- Shared performance insights for tuning production environments.
- Assisted in identifying optimal database on test outcomes.
- Enabled informed decision-making for infrastructure scalability and efficiency.

- Identified the most scalable and performant database solution tailored to service needs.
- Enabled data-driven decisions for DB & backend service architecture.
- Boosted backend efficiency by 25% through systematic benchmarking and tuning of database configurations under simulated production loads.

Report-Volume Load Testing

Problem Statement: Bulk report generation caused timeouts and high resource usage during large datasets or concurrent user requests.

Tech: RestAssured, Java, Selenium

Interface: Database + Backend Service

Role & Contributions:

- Ingested transactions using APIs to simulate peak and worst-case data.
- Simulated user flows on UI to assess report generation under realistic conditions.
- Measured end-user perceived performance during high-load report interactions.
- Captured BE/DB memory, response time, and system throughput under volume stress.
- Identified report generation bottlenecks through load and stress testing.

- Enabled large dataset handling and concurrent ingestion, reducing report-related testing efforts by 90%.
- Delivered performance insights to optimize backend logic and enhance responsiveness for business-critical reporting.
- Assisted report reliability under production load by identifying backend / database bottlenecks affecting peak-time performance.