OLP – Performance Testing Documentation

**OLP – Performance Testing & Optimization Documentation**

Performance testing was conducted to ensure that the OLP (Online Learning Platform) operates efficiently under expected user loads and delivers a smooth user experience across all devices and network conditions.

**1. API Load Testing**

* **Tools Used:** Postman (for functional test suites), Apache JMeter (for concurrent load simulation).
* **Objective:** Measure API reliability and response time when handling simultaneous requests from multiple users (e.g., student logins, course enrollments, certificate generation).
* **Result:**
  + 95th percentile response time: **< 300ms**
  + 1000 concurrent users handled without downtime or API failure.
* **Stress Test Outcome:** System showed stable behavior up to 1500 concurrent hits with only slight latency increases.

**2. UI Performance Testing**

* **Devices Tested:** Android (low/mid/high-end), iPhone, Desktop (Windows/Linux).
* **Network Conditions Simulated:** 4G, 3G, and limited bandwidth scenarios using Chrome DevTools.
* **Findings:**
  + Average page load time: **< 2 seconds**
  + Core interactions (login, course start, video playback): < 1.5s
* **Tools Used:** Lighthouse, GTmetrix, and WebPageTest.

**3. Database Performance & Data Handling**

* **Database:** MongoDB Atlas (cloud-hosted NoSQL).
* **Tests Conducted:**
  + Bulk insertions of student records (up to 50k entries).
  + Search/query response for courses, certificates, and profile updates.
* **Performance Metrics:**
  + Query time under normal load: **< 100ms**
  + Aggregation queries (for certificates & progress tracking): **< 250ms**
* **Indexes Added:** On frequently queried fields like userID, courseID, and email.

**4. Optimization Outcomes**

* **Current Results:**
  + API Response Time (avg.): **< 300ms**
  + UI Rendering Time: **< 2s** on 90% of tested devices.
  + Backend CPU & Memory usage remained below 70% during peak testing.

**5. Future Optimization Plans**

To further enhance performance and reduce load times:

* ✅ **Lazy Loading** – Load course content and images only when needed.
* ✅ **Content Compression** – Use GZIP/Brotli to compress responses.
* ✅ **Caching Strategies** – Store frequently accessed resources (e.g., user data, course thumbnails) on client/browser and CDN.
* ✅ **Database Sharding** – Prepare for large-scale deployments by horizontally partitioning data.
* ✅ **CDN Integration** – Host static assets via CDN to reduce latency globally.