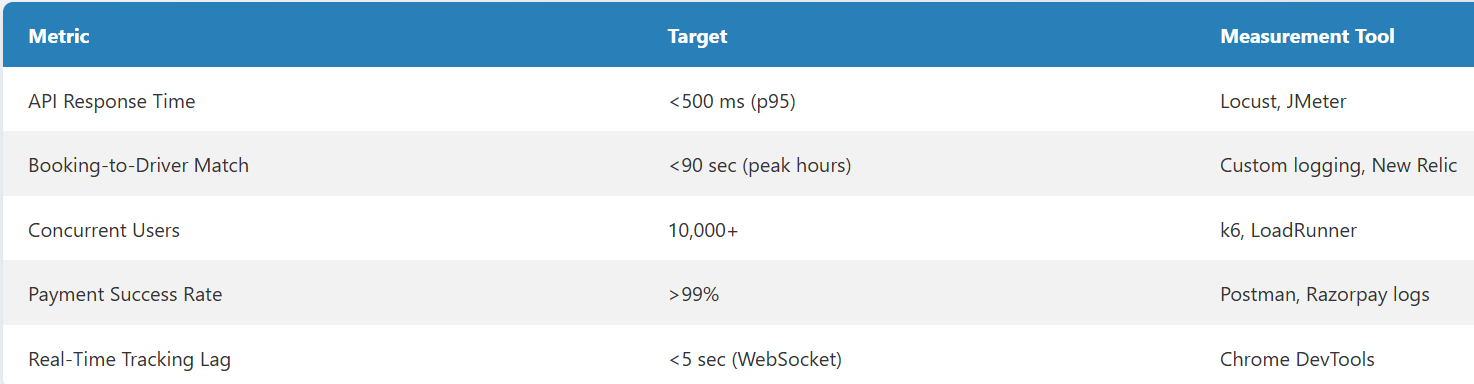
**Performance Testing**

**Model Performance Test**

|  |  |
| --- | --- |
| Date | 14 APRIL 2025 |
| Team ID | SWTID1742640402 |
| Project Name | MyRide |
| Maximum Marks | 4 Marks |

**1. Key Performance Metrics**

****

**2. Test Scenarios**

**a. Load Testing**

* Simulate **10,000 users** booking rides simultaneously.
* Measure:
  + API latency (POST /api/bookings).
  + Database query performance (MongoDB Atlas metrics).

**b. Stress Testing**

* Spike to **20,000 users** during "peak hour" simulation.
* Monitor:
  + Node.js server CPU/RAM usage (AWS CloudWatch).
  + Auto-scaling triggers (if using Kubernetes).

**c. Endurance Testing**

* Sustain **5,000 active users** for 24 hours.
* Check for:
  + Memory leaks (Node.js heap dumps).
  + Database connection pool exhaustion.

**d. Real-Time Tracking Test**

* **100 drivers** emitting location updates every 10s via Socket.io.
* Validate:
  + Passenger app reflects updates with <5s lag.
  + WebSocket server handles bursts (e.g., traffic jams).

**e. Payment Gateway Reliability**

* Process **1,000 transactions/minute** with Razorpay/Stripe.
* Track:
  + Failed transactions (retry logic efficacy).
  + HTTPS handshake time.

**3. Tools and Setup**



**4. Test Data Preparation**

* **Geospatial Data**:
  + Seed MongoDB with **10,000 drivers** (lat/lng spread across a city).
* **User Profiles**:
  + Generate **50,000 test passengers** (diverse locations, payment methods).

**5. Pass/Fail Criteria**

* **Pass**: all test cases passed

**6. Optimization Strategies**

* **Database**: Add compound indexes (location\_2dsphere, booking\_status).
* **Caching**: Redis for frequent queries (e.g., "nearby drivers").
* **CDN**: Offload static assets (maps, UI files).