**Project Development Phase**

**Model Performance Test**

|  |  |
| --- | --- |
| **Date** | 25 JUNE2025 |
| **Team ID** | LTVIP2025TMID31613 |
| **Project Name** | Garage Management System |
| **Maximum Marks** |  |

**Model Performance Testing:**

To test model performance in a Garage Management System (GMS), you'd typically be evaluating how well your system handles tasks like appointment scheduling, inventory tracking, billing, and customer management—especially if you're using machine learning or rule-based automation. Here's how you can approachit**:**

**Key Areas of Performance Testing**

* **Response Time**: Measure how quickly the system responds to user actions (e.g., booking a service or generating an invoice).
* **Throughput**: Test how many transactions or service requests the system can handle per second/minute**.**
* **Scalability**: Simulate increased load (more users, more vehicles) to see how the system scales.
* **Resource Utilization:** Monitor CPU, memory, and database usage under different loads.
* **Error Rate:** Track how often the system fails or returns incorrect results under stress.

**You're Using AI/ML Models:**

If your GMS includes predictive models (e.g., for estimating service time or predicting part failures), you’ll want to test:

* Accuracy: How close are the predictions to actual outcomes?
* Precision/Recall: Especially important if you're flagging potential issues like part failures.
* Latency: How fast does the model return results?
* Model Drift: Is the model still accurate over time as new data comes in?

**Tools You Can Use**

* Apache JMeter or Locust for load and stress testing
* Postman for API performance testing
* TensorBoard or MLflow for tracking ML model metrics
* New Relic or Datadog for real-time system monitoring

**Functional & Performance Testing**

**Model Performance Test**

|  |  |
| --- | --- |
| **Date** | 25 JUNE 2025 |
| **Team ID** | LTVIP2025TMID31613 |
| **Project Name** | Garage Management System |
| **Maximum Marks** |  |

**Test Scenarios &Results:**

**Functional Testing:**

This ensures each feature behaves as intended:

* User Authentication: Login/logout, role-based access (admin, mechanic, customer)
* Appointment Scheduling: Booking, rescheduling, and cancellation flows
* Inventory Management: Adding, updating, and tracking parts
* Billing & Invoicing: Generating accurate bills, applying discounts, tax calculations
* Service History Tracking: Recording and retrieving vehicle service records
* Notifications: Email/SMS alerts for service reminders or updates

**Performance Testing** :

|  |
| --- |
|  |

|  |  |  |
| --- | --- | --- |
| **Scenario** | **What to Test** | **Performance Metric** |
| Booking surge | 100+ users book services simultaneously | Response time < 2s |
| Inventory update | Bulk upload of 500+ parts | CPU/memory usage, DB latency |
| Invoice generation | Generate 1000 invoices in batch | Throughput (invoices/min) |
| Search service history | Query 10 years of records | Query time < 1s |

**Tools:**

* **Selenium:** Automate functional test cases
* **JMeter or Locust:** Simulate concurrent users and load
* **Postman:** Test API endpoints for correctness and speed
* **Grafana + Prometheus:** Monitor system metrics in real time