# **Functional & Performance Testing Template**

**Model Performance Test**

|  |  |
| --- | --- |
| Date | 26 June 2025 |
| Team ID | LTVIP2025TMID37089 |
| Project Name | Sustainable Smart City Assistant using IBM Granite LLM |
| Maximum Marks |  |

## **Test Scenarios & Results**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Scenario (What to test)** | **Test Steps (How to test)** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| **FT-01** | Input Box Text Validation (Eco queries, complaints, prompts) | Enter valid text like "How to save water?" and submit. Then try blank and special character inputs and observe behavior. | Valid inputs should return relevant responses. Invalid inputs should trigger error messages like "Please enter valid query." | Valid queries returned accurate, useful outputs. | Pass |
| **FT-02** | Complaint Classification & Routing | Enter complaints like “Street lights not working” or “Water leakage nearby” and submit. Check if the assistant routes each to the correct department. | Each complaint should be identified and tagged (e.g., Electricity, Water Board). The response should mention the appropriate department. | Complaints were correctly classified based on keywords. | Pass |
| **FT-03** | Eco-Query Generation | Enter queries like “How to reduce pollution in my area?” or “Best way to save electricity?” Click **Generate** and review the assistant’s response. | The assistant should provide relevant, sustainable tips or strategies. Response must align with environmental concerns using LLM knowledge. | Model generated accurate and actionable eco-advice. | Pass |
| **FT-04** | Real-time Response via LLM | Submit a civic or eco-related prompt like “Report a drainage issue” or “Ways to manage waste.” Measure the response time using a timer or stopwatch. | Response should be generated within 3 seconds. Content must be relevant and clearly address the user’s input. | Responses were generated in 2.1 to 2.4 seconds on average. | Pass |
| **PT-01** | Simultaneous User Handling | Open multiple browser tabs and submit different queries at the same time. Observe how the system handles multipe parallel requests. | All queries should be processed without crashing or significant delay. Each tab should receive its own correct and timely response. | All requests were handled successfully with slight delay (~2.8s). | Pass |
| **PT-02** | Error Handling (Invalid API use or broken model load) | Simulate API failure by renaming model path or breaking API call in code. Run the app and try submitting a query. | System should not crash; it must display an error like “Model load failed.” App should remain stable and guide user to retry or contact admin. | System showed a clear error message: “Model failed to load. Please try again later.” | Pass |
| **PT-03** | Frontend UI Responsiveness | Access the application on desktop, tablet, and mobile devices. Interact with UI elements like buttons, input fields, and mode switches. | UI should adapt to all screen sizes and remain fully functional. Buttons, inputs, and outputs must remain visible and usable. | UI worked well across devices with responsive layout. | Pass |