6. FUNCTIONAL AND PERFORMANCE TESTING

The **Smart City Assistant** application underwent rigorous functional and performance testing across all integrated modules to ensure reliability, responsiveness, and user experience consistency. Each feature was tested independently as well as in combination, simulating realistic user interactions and data flows.

6.1 Functional Testing

Functional testing was carried out on nine modules:

- **Weather Forecast**
  Verified accurate retrieval of current and weekly weather data using the OpenWeatherMap API. Edge cases like invalid city names were handled gracefully with error messages.

- **Air Pollution Monitoring**
  API response was tested for various cities globally. The module dynamically displayed AQI values and pollutant breakdowns. Accuracy was cross-checked with official sources.

- **Traffic Monitoring**
  The TomTom API was tested for real-time traffic updates. Latency, congestion levels, and traffic flows were plotted effectively on demand.

- **Policy Summarizer**
  PDF files of various formats were uploaded to evaluate NLP processing. Summaries were compared with manual human-written versions to ensure coherence and relevance.

- **Chat Assistant**
  Functionality was validated with diverse question types, including policy-related, general knowledge, and contextual follow-ups. Integration with LLM was verified for both IBM Watsonx and OpenAI endpoints.

- **KPI Forecasting**
  Uploaded historical usage data in CSV format was correctly parsed and forecasted using the Prophet library. Forecasts were plotted, validated, and exported.

- **Anomaly Detection**
  Simulated usage datasets were tested for outlier detection. The algorithm successfully flagged irregular patterns in electricity usage beyond thresholds.

- **Customer Feedback Form**
  Tested submission, storage, and CSV export. Form validation prevented empty

submissions and stored data reliably.

- **Eco Tips Module**
  Ensured correct rendering of tips based on environmental context (e.g., air quality, traffic, energy conservation).

Edge testing included:

- No internet / API failure
- Missing `.env` keys
- Uploading unsupported file formats
- Empty inputs in forms and chatbot

All modules responded with appropriate user feedback and error-handling mechanisms.

6.2 Performance Testing

Performance was evaluated on a standard system (Intel i5, 8GB RAM) running Python 3.11 and Streamlit.

Module	Response Time	Remarks
Weather Forecast	1.2 sec	Dependent on OpenWeatherMap latency
Air Pollution	1.5 sec	Consistent small payload
Traffic Monitoring	2.1 sec	TomTom API delays under high-load testing
Policy Summarizer	4.8 sec	LLM inference time on Watsonx
Chatbot	3.2 sec	Varies by API (OpenAI ~2.8s,Watsonx ~3.6s)
KPI Forecasting	5.0 sec	Prophet training on 365+ days of data
Anomaly Detection	1.4 sec	Lightweight ML-based analysis
Feedback Form	<1 sec	Instant CSV write and UI update
Eco Tips	<1 sec	Static tips rendering

All API requests were asynchronous and fault-tolerant. The interface remained stable even during multi-module interactions. Streamlit handled concurrent updates without freezing or crashing.