RoostGPT Input and Output Table

Overview

RoostGPT is an intelligent test automation platform that leverages AI to transform business requirements and technical specifications into comprehensive test suites across multiple testing frameworks.

Test Types

Quick Reference Table

Test Type	Input	Output
Unit Test	• Source Code (Java, Python, Golang, CSharp)	• Test Code (Java, Python, Golang, CSharp)
API Test	• Git Repo (for output) • Swagger (OpenAPI spec)	• Postman • Rest-Assured • Artillery • Karate • Pytest (one of them) • Test Data (json)
Functional Test	• Jira User Story (ID or file) • User Input (file or text)	• JSON output • Gherkin Feature File • Functional Test excel output • OpenAPI Spec

Test Type	Input	Output
UI Test	• Domain (url for which test need to be generated) • User Scenario Document, • Login Credentials (if applicable) • Login Scenario Document	• JS Playwright Test Script

Unit Test

Purpose: Unit tests validate individual components, functions, or methods in isolation, ensuring they behave correctly under various conditions.

Input Requirements:

- □ Source code files in supported languages (Java, Python, Golang, or C#)
- □ Code should be well-structured with clear function/method definitions
- □ Dependencies and imports should be properly declared

Output Generated:

- □□ Comprehensive test code in the same language as the source
- □□ Test cases covering normal operations, edge cases, and error conditions

□□ Appropriate assertions and test data

API Test

Purpose: API tests verify the functionality, reliability, and performance of application programming interfaces, ensuring they meet specifications and handle requests correctly.

Input Requirements:

- □ Git repository URL for storing generated test artifacts
- □ OpenAPI/Swagger specification document defining:
  - API endpoints
  - Request/response schemas
  - Authentication requirements

Output Generated:

Test collections and scripts in your choice of framework:

Framework	Description	Output Format
Postman	Collection JSON files ready to import	`.json`
Rest-Assured	Java-based test classes with fluent API syntax	`.java`
Artillery	YAML configuration for load and performance testing	`.yml`
Karate	Feature files with BDD-style API tests	`.feature`
Pytest	Python test functions with request fixtures	`.py`

Functional Test

Purpose: Functional tests validate complete business workflows and user scenarios, ensuring the system behaves according to specified requirements and user expectations.

Input Requirements:

- **Jira User Story:** Can be provided as a Jira ticket ID or exported file
- **User Input:** Business requirements as text or document files describing expected system behavior

Output Generated:

- **JSON Output:** Structured test data and results in JSON format
- **Gherkin Feature Files:** BDD-style scenarios in Given-When-Then format
- **Excel Output:** Test case documentation with steps and expected results

OpenAPI Spec: Generated API specifications for tested endpoints

UI Test

Purpose: UI tests automate user interactions with web applications, verifying that user interface elements function correctly and the application responds appropriately to user actions.

Input Requirements:

- **Domain URL:** The base URL of the application to be tested
- **User Scenario Document:** Detailed description of user workflows and expected interactions
- **Login Credentials:** Authentication details if the application requires login (optional)
- **Login Scenario Document:** Step-by-step login process if authentication is required

Output Generated:

- **Playwright Test Scripts:** JavaScript test files using Playwright framework
- Automated browser interactions including clicks, form fills, and navigation
- Assertions for verifying page elements, content, and behavior

---

Revision #3
Created 9 November 2025 14:40:26 by Divyesh
Updated 9 November 2025 15:05:55 by Divyesh