RoostGPT Test Generation

This Document give info about the AI providers which is supported in RoostGPT for different test types.

RoostGPT AI Provider Support for UI Test Generation

Overview

RoostGPT supports three major AI providers for UI test generation:

- **OpenAI**
- **Google Gemini**
- **Azure OpenAI**

All models support vision capabilities for analyzing UI elements and generating comprehensive test cases.

Note - RoostGPT only support vision models for UI Test Generation.

OpenAI Models

Model Name	Context Window	Best For	Capabilities
GPT-5	Extended	Complex UI workflows, enterprise applications	Expert-level intelligence, built-in reasoning, superior visual perception
GPT-4.5	Large	Standard UI testing	Vision-enabled, balanced performance
GPT-4o (Omni)	Large	Multimodal test generation	Real-time visual analysis, fast response
GPT-4o mini	Standard	High-volume test generation	Cost-efficient, fast, good vision

Model Name	Context Window	Best For	Capabilities
o3	Extended	Complex test logic, deep reasoning	Advanced chain-of-thought with vision
o4-mini	Standard	Cost-efficient reasoning	Faster reasoning with vision support

Google Gemini Models

Model Name	Context Window	Best For	Capabilities
Gemini 2.5 Pro	Input: 1024k Output: 64k	Complex UI testing, large-scale apps	State-of-the-art thinking, long context
Gemini 2.5 Flash	Input: 1024k Output: 64k	Fast, cost-effective test generation	Best price-performance, agentic use cases
Gemini 2.5 Flash-Lite	Input: 1024k Output: 64k	High-volume, low-latency generation	Most cost-effective, optimized for throughput
Gemini 2.0 Flash	Input: 1024k Output: 8k	Next-gen features, modern apps	Native tool use, improved speed
Gemini 2.0 Flash-Lite	Input: 1024k Output: 8k	Budget-conscious projects	Cost-efficient, low latency

Azure OpenAI Models

Azure OpenAI provides the same OpenAI models with enterprise features.

Model Name	Context Window	Additional Features
GPT-5 Series	Extended	Azure security, compliance, VNET integration
GPT-4.5	Large	Private endpoints, managed identity
GPT-4.1 Series	Large	Regional deployment, data residency
GPT-4o Series	Large	SLA guarantees, Azure Monitor integration
o3, o4-mini	Extended/Standard	Azure AD authentication

Recommended Models

OpenAI

- **gpt-4o**
- **gpt-5**

Azure OpenAI

- **gpt-4o**
- **gpt-5**

Google Gemini

- **gemini-2.5-pro**
- **gemini-2.5-flash**

Support Resources

- **OpenAI**: platform.openai.com/docs
- **Google Gemini**: ai.google.dev/gemini-api/docs
- **Azure OpenAI**: learn.microsoft.com/azure/ai-services/openai

RoostGPT AI Provider Support for Unit, API and Other Tests

Overview

RoostGPT supports multiple AI providers for comprehensive test generation across unit tests, API tests, Integration Tests, Functional Tests and more. This document outlines the supported providers and their available models.

Supported Providers:

- OpenAI
- Claude AI (Anthropic)
- Azure OpenAI
- Vertex AI (Google Cloud)
- AWS Bedrock

OpenAI Models

Model Name	Context Window (Tokens)	Best For	Key Features
GPT-5	4 Million Input: 272k Output: 128k	Complex test scenarios, enterprise	Expert-level intelligence, built-in reasoning
GPT-4.5	Large	Standard test generation	Balanced performance, cost-effective
GPT-4o	Large	Fast test generation	Multimodal, real-time processing
GPT-4o mini	Standard	High-volume generation	Cost-efficient, fast
GPT-4 Turbo	128K tokens	Production environments	Proven reliability

Model Name	Context Window (Tokens)	Best For	Key Features
o3	Extended	Complex reasoning tasks	Advanced chain-of-thought
o4-mini	Standard	Cost-efficient reasoning	Faster reasoning model

Claude AI (Anthropic) Models

Model Name	Context Window (Tokens)	Best For	Key Features
Claude Opus 4.1	Input: 200K Output: 32k	Complex coding tasks, unit tests	Best coding model, 74.5% on SWE-bench
Claude Sonnet 4.5	Output: 64k	Coding, Finance and CyberSecurity	Long running tasks, enhanced domain knowledge and 77.2% on SWE-bench
Claude Sonnet 4	200K-1M Output: 64k	General test generation	Superior coding and reasoning, instruction following
Claude Sonnet 3.7	200K tokens Output: 64k	Hybrid reasoning tasks	Fast and extended thinking modes
Claude Sonnet 3.5	200K tokens Output: 8k	Legacy support	Strong coding capabilities
Claude Haiku 3.5	200K tokens Output: 8k	Quick, simple tests	Fast, cost-effective

Azure OpenAI Models

Azure OpenAI provides the same OpenAI models with enterprise features.

Model Name	Context Window	Additional Features
GPT-5 Series	4 Million Input: 272k Output: 128k	VNET integration, Private endpoints
GPT-4.5	Large	Managed Identity, Regional deployment
GPT-4o Series	Large	SLA guarantees, Azure Monitor

Model Name	Context Window	Additional Features
GPT-4 Turbo	128K tokens	Standard and Provisioned deployments
o3, o4-mini	Extended/Standard	Azure AD authentication

Vertex AI (Google Cloud) Models

Model Name	Context Window (Tokens)	Best For	Key Features
Gemini 2.5 Pro	Input: 1024k Output: 64k	Complex API testing, large codebases	Deep Think mode, long context

---

AWS Bedrock Models

AWS Bedrock provides access to multiple foundation models from various providers in a fully managed service.

Available Model Families

Provider	Models	Context Window	Best For
Anthropic	Claude Opus 4.1, Sonnet 4, Sonnet 3.7	Output Token - 4K	Code generation, unit tests

---

Recommended Models

OpenAI

- GPT-4o, GPT-5

Azure OpenAI

- GPT-4o, GPT-5

Claude

- Claude Sonnet 4.1

Support Resources

- **OpenAI**: platform.openai.com/docs
- **Anthropic (Claude)**: docs.claude.com
- **Azure OpenAI**: learn.microsoft.com/azure/ai-services/openai
- **Vertex AI**: cloud.google.com/vertex-ai/docs
- **AWS Bedrock**: docs.aws.amazon.com/bedrock

RoostGPT Test Generation - Prerequisites

Overview

Before using RoostGPT for test generation, ensure you have the following prerequisites based on your test type.

---

1. Git Integration (For PR Output)

When needed: If test output should be created as a Git Pull Request

Required:

- Git Personal Access Token with Read, Write, and Create PR permissions
- Repository URL and target branch name

Support Link: https://docs.roost.ai/books/git-configuration-and-tokens

---

2. AI Provider Access Token

When needed: For all test generation

Required:

- API Key/Access Token from one of the supported providers:
  - OpenAI
  - Claude AI (Anthropic)
  - Azure OpenAI (API Key + Endpoint)
  - Vertex AI (Google Cloud)
  - AWS Bedrock

- Deepseek

Support Link: https://docs.roost.ai/books/ai-configuration-and-tokens

---

3. Jira Integration (For Functional Tests)

When needed: Generating functional tests from Jira tickets

Required:

- Jira Access Token
- Jira Domain URL (e.g., `https://company.atlassian.net`)

---

4. UI Test Generation - Login Credentials

When needed: Generating UI/E2E tests requiring authentication

Required:

- Application Login Username
- Application Login Password

---

5. UI Test Generation - Scenario Document

When needed: Testing specific user workflows or scenarios

Required:

- Document containing scenario details (PDF, Markdown, or Text format)
- Document should include:
  - Scenario name
  - Test steps
  - Expected results
  - Test data

---

6. UI Test Generation - Custom Login Steps Document

When needed: Application has non-standard login flows (SSO, OAuth, custom authentication)

Note: MFA (Multi-Factor Authentication) is not supported

Required:

- Document describing custom login steps
- Document should include:
  - Login method type
  - General authentication process

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

RoostGPT AI Keys Requirements

AI Configuration Keys

This document provides the required environment variables for configuring different AI providers.

---

1. OpenAI Configuration

Required Environment Variables

Variable Name	Required	Description
`OPENAI_API_KEY`	Yes	Your OpenAI API key
`OPENAI_API_MODEL`	Yes	Model to use

Configuration Example

```
OPENAI_API_KEY=sk-proj-xxxxxxxxxxxxxxxxxxxx
OPENAI_API_MODEL=gpt-4o
```

---

2. Google Gemini Configuration

Required Environment Variables

Variable Name	Required	Description
`GEMINI_API_KEY`	Yes	Your Google AI Studio API key for accessing Gemini models
`GEMINI_MODEL`	Yes	Gemini model to use. Options: `gemini-pro`, `gemini-pro-vision`, `gemini-ultra`, etc.

Configuration Example

```
GEMINI_API_KEY=AIzaSyxxxxxxxxxxxxxxxxxxxxxxxxxxxx
GEMINI_MODEL=gemini-pro
```

---

3. AWS Bedrock Configuration

Required Environment Variables

Variable Name	Required	Description
`AWS_BEDROCK_MODEL`	Yes	Model ID to use. Examples: `anthropic.claude-v2`, `anthropic.claude-3-sonnet-20240229-v1:0`, `amazon.titan-text-express-v1`
`AWS_DEFAULT_REGION`	Yes	AWS region where Bedrock is available. Examples: `us-east-1`, `us-west-2`, `eu-west-1`
`AWS_ACCESS_KEY_ID`	Yes	AWS access key ID for authentication
`AWS_SECRET_ACCESS_KEY`	Yes	AWS secret access key for authentication

Configuration Example

```
AWS_BEDROCK_MODEL=anthropic.claude-3-sonnet-20240229-v1:0
AWS_DEFAULT_REGION=us-east-1
AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

Important Notes

- Model availability varies by region. Check AWS Bedrock documentation for supported models in your region.
- You may need to request model access through the AWS Bedrock console before using certain models.
- Ensure your IAM user has necessary permissions for `bedrock:InvokeModel` action.

---

4. Claude AI Configuration

Required Environment Variables

Variable Name	Required	Description
`CLAUDE_AI_API_KEY`	Yes	Your Anthropic API key for accessing Claude models
`CLAUDE_AI_MODEL`	Yes	Claude model to use. Examples: `claude-3-opus-20240229`, `claude-3-sonnet-20240229`, `claude-3-haiku-20240307`, `claude-2.1`, `claude-2.0`

Configuration Example

```
CLAUDE_AI_API_KEY=sk-ant-api03-xxxxxxxxxxxxxxxxxxxxxxxxxx
CLAUDE_AI_MODEL=claude-3-sonnet-20240229
```

---

5. Azure OpenAI Configuration

Required Environment Variables

Variable Name	Required	Description
`AZURE_OPENAI_ENDPOINT`	Yes	Your Azure OpenAI resource endpoint URL (e.g., `https://your-resource.openai.azure.com/`)

Variable Name	Required	Description
`AZURE_DEPLOYMENT_NAME`	Yes	Name of your deployed model in Azure OpenAI Studio (e.g., `gpt-4-deployment`, `gpt-35-turbo-deployment`)
`AZURE_OPENAI_KEY`	Yes	API key for your Azure OpenAI resource (Key 1 or Key 2 from Azure portal)

Configuration Example

```
AZURE_OPENAI_ENDPOINT=https://your-resource-name.openai.azure.com/
AZURE_DEPLOYMENT_NAME=gpt-4-deployment
AZURE_OPENAI_KEY=1234567890abcdef1234567890abcdef
```

Important Notes

- Azure OpenAI requires approval. Apply for access if you haven't already.
- Model availability varies by region. Choose your region based on available models.
- Deployment names are custom - you choose them when deploying models in Azure OpenAI Studio.
- The endpoint URL should end with a trailing slash (`/`).

Manual Playwright Chromium Browser Installation Guide

This guide provides step-by-step instructions for manually downloading and setting up the Playwright Chromium browser on Windows.

Prerequisites

- Windows operating system
- File extraction tool (e.g., Windows Explorer, 7-Zip, WinRAR)
- Use Powershell to run commands.

Installation Steps

Step 1: Download the Browser

Download the Chromium browser archive from the following URL:

```
https://drive.google.com/drive/u/0/folders/1xcmG-rfTcWzf7DEuNt3DUTpwe7FROhTq

OR

https://playwright-verizon.azureedge.net/builds/chromium/1187/chromium-win64.zip
```

Step 2: Create the Playwright Directory

Navigate to your user profile's AppData folder and create the ms-playwright directory:

```
%USERPROFILE%\AppData\Local\ms-playwright
```

You can do this via this command:

```
mkdir "%USERPROFILE%\AppData\Local\ms-playwright"
```

Step 3: Create the Browser Version Folder

Inside the `ms-playwright` folder, create a new folder named `chromium-1187`:

```
mkdir "%USERPROFILE%\AppData\Local\ms-playwright\chromium-1187"
```

Step 4: Extract the Browser

Unzip the downloaded `chromium-win64.zip` file into the `chromium-1187` folder.

After extraction, your folder structure should look like:

```
%USERPROFILE%\AppData\Local\ms-playwright\
└── chromium-1187\
    └── chrome-win\
        ├── chrome.exe
```

Step 5: Create Validation Files

Create two empty marker files inside the `chromium-1187` folder to indicate successful installation:

File 1: `DEPENDENCIES_VALIDATED`

```
ni "%USERPROFILE%\AppData\Local\ms-playwright\chromium-1187\DEPENDENCIES_VALIDATED"
```

File 2: `INSTALLATION_COMPLETE`

```
ni "%USERPROFILE%\AppData\Local\ms-playwright\chromium-1187\INSTALLATION_COMPLETE"
```

Final Directory Structure

After completing all steps, your directory structure should be:

```
%USERPROFILE%\AppData\Local\ms-playwright\
└── chromium-1187\
    ├── chrome-win\
    │   ├── chrome.exe
    ├── DEPENDENCIES_VALIDATED
    └── INSTALLATION_COMPLETE
```

Verification

Check Files and Folders

To verify the installation, check that all files and folders exist:

```
dir "%USERPROFILE%\AppData\Local\ms-playwright\chromium-1187"
```

You should see the `chrome-win` folder along with the two marker files.

Test Browser Launch

To verify that Playwright is correctly picking up the browser path, run the following command to open a URL:

```
playwright open https://www.google.com
```

This should launch a new Chromium browser window and navigate to the specified URL. If the browser opens successfully, your manual installation is working correctly.

Troubleshooting

- **Browser not detected:** Ensure the folder name is exactly `chromium-1187` and both marker files exist.
- **Extraction issues:** Make sure the `chrome-win` folder is directly inside `chromium-1187`, not nested in an additional folder.
- **Permission errors:** Run Powershell as Administrator if you encounter access issues.