# **AI in QA Starter Playbook**

# Pattern of wrenches on a yellow background.

This guide is designed to help you get familiar with the AI tools we use to enhance our QA processes.

Don’t worry if it feels like a lot at first — with practice and hands-on experience, you’ll quickly get comfortable and see the impact these tools can make.

Let's get started!

**Last Updated Date : 30 April 2025**

## **What Are These AI Tools For?**

At Techjays, we use AI tools to:

* Understand user requirements
* Create test cases faster
* Review Figma designs
* Write better bug reports
* Help automate various testing tasks
* Make our documentation more consistent

Remember: These tools are here to help you, not replace your skills! They're usually about 80-90% accurate, so always review what they generate.

## **Your AI Toolkit: Quick Overview**

| **Tool** | **Best For** | **When To Use It** |
| --- | --- | --- |
| Claude | Complex scenarios, image analysis | When you need detailed test cases or to review designs |
| ChatGPT | Quick ideas, simple tests | When you need fast results for straightforward tasks |
| Testara | Test automation | When you need to automate repetitive testing |
| Grok | Creative approaches | When you want alternative perspectives |
| Gemini | Google ecosystem integration | When working with Google tools |

Note on Claude Models:

* We primarily use Claude 3.7 Sonnet for most QA tasks
* For more complex reasoning or precision tasks, we use Claude 3 Opus (we call it "o3" in our team)

|  |
| --- |

## **Getting Started: First Steps**

### 1. Request Access

* Ask your team lead or IT for access to Claude and ChatGPT
* For Testara, use our company instance at <https://testara.techjays.com>
* Note: Mac users might experience some compatibility issues

### 2. Choose the Right Tool

* Different tools work better for different tasks (see the table above)
* For most tasks, we recommend starting with Claude

### 3. Be Mindful of Privacy

* Don't share sensitive client information with these AI tools

## **How to Use AI for Common QA Tasks**

### **Creating Test Cases**

**Simple Example Prompt:**

For the user story [paste story here], create test cases covering:

1. Positive scenarios

2. Negative scenarios

3. Edge cases

Format each test case with:

- Test ID

- Prerequisites

- Test steps

- Expected results  
 - Actual results

**Tips:**

* Be specific about what you want
* Include relevant requirements
* Ask for a specific format

### **Reviewing UI Designs**

Simple Example Prompt:

Review this Figma design screenshot from a QA perspective. Check for:

1. UI consistency issues

2. Alignment problems

3. Potential usability issues

4. Does it match the requirements?

**Tips:**

* Upload clear, high-quality screenshots
* Mention specific areas to focus on
* Include relevant requirements for context

### **Writing Bug Reports**

**Simple Example Prompt:**

Create a detailed bug report for this issue:

[describe what you found]

Format it with:

1. Title

2. Description

3. Steps to reproduce

4. Expected result

5. Actual result

6. Priority (Critical/High/Medium/Low)

**Tips:**

* Include all relevant details about the issue
* Mention the environment (browser, device, etc.)
* Be specific about steps to reproduce

## **Setting Up Advanced Tools**

### **Claude MCP for Test Automation**

MCP (Model Context Protocol) lets Claude interact with web applications to help with automated testing.

**What is MCP?** Model Context Protocol (MCP) is a standard that connects AI assistants to various systems and tools. For QA, this means Claude can directly interact with web applications for automated testing.

**Basic Installation Steps:**

1. **Install the Claude Desktop Application**  
   * Open the Claude application
   * Navigate to File -> Settings -> Developer
   * Click on Edit Config
2. **Add MCP Configuration**  
   * This will open the Claude folder
   * Open the JSON configuration file
   * Paste the following configuration:

{

"mcpServers": {

"puppeteer": {

"command": "npx",

"args": ["@modelcontextprotocol/server-puppeteer"]

},

"sequential-thinking": {

"command": "npx",

"args": ["@modelcontextprotocol/server-sequential-thinking"]

}

}

}

1. **Save the file**
2. **Install Required Packages**  
   * Run these commands in your terminal:

npm install -g @modelcontextprotocol/server-puppeteer

npm install @modelcontextprotocol/server-sequential-thinking

npm install -g @modelcontextprotocol/server-sequential-thinking

1. **Troubleshooting Compatibility Issues**  
   * If you encounter compatibility problems, try:

npx -y @modelcontextprotocol/server-puppeteer@latest

*For additional help with setup, check with your team lead or the IT department.*

Example Usage:

Navigate to https://example.com/login and:

1. Enter "testuser" in the username field

2. Enter "password123" in the password field

3. Click the Login button

4. Verify the dashboard appears

## **Testara Setup**

Testara is our own specialized QA automation tool. Here's a simplified setup guide:

**Step 1: Install Required Dependencies**

* Python (3.12.x or later)
  + During installation, check "Add Python to PATH"
* Node.js (LTS version)
  + For macOS users: **brew install node**

**Step 2: Verify Installation**

* Run these commands to check:  
   python --versionnode --versionnpm --versionnpx --version

**Step 3: Install Required Packages**

* Install uv package manager:  
   pip install uv
* Install Playwright MCP Server:  
   npm install -g @executeautomation/playwright-mcp-server

**Step 4: Setup Desktop App**

* Download the 5ire Desktop App (ask your team lead for the link)
* Create folder structure: Create a "test" folder and a "data" folder inside it
* Configure the app by adding:
  + Playwright tool
  + Sequential Thinking tool
  + Google API key (obtained from Google AI Studio)

For complete setup instructions with screenshots, please refer to the [detailed documentation](https://docs.google.com/document/u/0/d/1tgGlSSSc56EEHMxCnOGW-m5rGEIyMjh3SZGW2nGtECw/edit) or ask your team lead for assistance.

## **Tips for Success**

### **The Golden Rule: Context is Everything**

The most important factor for getting good results from AI tools is providing good context:

* Be Clear and Specific: Vague prompts = vague results
* Give Examples: Show the format you want
* Iterate: Start simple, then refine based on results
* Include Requirements: Share the most important parts of requirements

### **Improving Your Results**

* Save Successful Prompts: Keep track of what works well
* Use Roles: Start with "Act as a QA engineer..." to get better responses
* Ask for Specific Formats: Tables, lists, or other structured formats
* Review Everything: Always check AI outputs for accuracy

### **Managing Knowledge & Documentation**

* Create template prompts for consistent formatting of test cases and bug reports
* Save successful prompts in a central repository for the team to reuse
* Organize test cases by feature, module, and test type
* Document limitations and workarounds for specific AI tools
* Use AI tools to standardize documentation across the team

### **Using Sequential Thinking for Complex Tests**

For complex test scenarios, you can use MCP's sequential thinking capability to break down multi-step processes into logical sequences. This helps when:

* Testing complex workflows that need several steps
* Situations where test steps depend on previous outcomes
* Cases that require careful validation at each step

Example prompt:

Use sequential thinking to test the checkout process:

1. First add items to cart

2. Proceed to checkout

3. Fill in shipping information

4. Validate payment options

5. Complete purchase

6. Verify order confirmation

## **Experimental & Emerging AI QA Tools**

Our team is exploring cutting-edge AI technologies for QA. Here's what's on our radar:

### **Self-Healing Test Automation**

AI that automatically updates locators or selectors when UI changes, reducing test maintenance.

* Example Tools: Testim, Functionize, Katalon, Mabl

### **Visual Testing & Computer Vision**

AI that intelligently compares screenshots, understanding context beyond pixel-by-pixel comparison.

* Example Tools: Applitools, Percy, Screener

### **Intelligent Test Case Generation**

Using NLP and LLMs to automatically generate test cases from requirements.

* Example Tools: TestRigor, Diffblue (we're already implementing this with our LLMs)

### **Exploratory Testing Assistance**

AI agents that suggest test scenarios and edge cases during exploratory testing.

* Example Tools: TestGPT-style assistants, ReTest AI

### **AI-Augmented Log Analysis**

AI tools that automatically analyze logs, detect anomalies, and summarize stack traces.

* Example Tools: Logz.io with AI, Elastic AI Assistant

### **Other Tools We're Experimenting With**

* TestRigor: For specific testing scenarios
* BugBug.io: For automated test creation/bug report
* Deepseek: For complex testing scenarios
* Notion AI: For test documentation
* Copilot: For test script development
* Google AI Studio: For custom model fine-tuning

## **Getting Help**

If you're stuck or have questions:

* Ask your team lead
* Check our internal knowledge base
* Share your experiences in our team meetings

Remember, everyone was new to these tools at some point. Don't hesitate to ask for help!

## **Quick Reference Card**

| **Task** | **Best Tool** | **Quick Prompt Template** |
| --- | --- | --- |
| Test Cases | Claude | "Create test cases for [feature] with these requirements: [paste]" |
| UI Review | Claude | "Review this Figma design for QA issues like alignment, consistency, and usability" |
| Bug Reports | Claude/ChatGPT | "Format this bug into a proper report: [describe issue]" |
| Test Automation | Claude MCP | "Navigate to [URL] and test the [feature] functionality" |
| Documentation | Claude | "Create documentation for [feature] in this format: [example]" |

|  |  |  |
| --- | --- | --- |

Welcome again to the team! We're excited to see how you'll use these tools to enhance our QA processes.

## **Last Updated Date : 30 April 2025**