**Simplita Testing Engine User Manual**

**1. Introduction**

The Simplita Testing Engine is a comprehensive automated testing solution designed to help users test web forms efficiently. This tool allows you to generate test cases, execute tests, analyze results, and generate reports - all from a user-friendly interface.

**2. Getting Started**

**2.1 Accessing the Testing Engine**

1. Navigate to the Simplita application
2. Click on the "Testing Engine" button in the main navigation bar
3. The Testing Engine interface will load with two main panels:

* Preview Panel (80% of the screen width)
* Chat Panel (20% of the screen width)

**2.2 System Requirements**

* Modern web browser (Chrome, Firefox, Safari, or Edge)
* Internet connection
* Access to the Simplita platform

**3. User Interface Overview**

**3.1 Preview Panel**

The Preview Panel displays:

* Form selection options
* Test results visualization
* Suggestions for failed test cases
* Chat history

**3.2 Chat Panel**

The Chat Panel allows you to:

* Communicate with the AI Testing Assistant
* Request test case generation
* Execute tests
* View test results
* Generate reports

**4. Core Features**

**4.1 Form Analysis**

The Testing Engine can analyze forms to identify:

* Form elements and structure
* Required fields
* Validation rules
* Submission endpoints

To analyze a form:

1. Select a form from the list
2. The system will automatically analyze the form structure
3. Analysis results will be used to generate appropriate test cases

**4.2 Test Case Generation**

**4.2.1 General Test Cases**

The system can generate comprehensive test cases covering:

* UI validation
* Input field validation
* Form functionality
* Error handling
* User experience

To generate general test cases:

1. Select a form
2. Click the "Generate Test Cases" button
3. The AI will analyze the form and generate appropriate test cases
4. Review the generated test cases

**4.2.2 Customized Test Scripts**

For advanced users, the system can generate custom Playwright test scripts:

1. Ask the AI Assistant to generate a Playwright test script
2. Provide details about the specific test scenario
3. The system will generate a complete Playwright test that follows best practices

**4.3 Test Execution**

Execute tests against your web forms:

1. Select a form
2. Choose the test cases to run
3. Toggle the "Headless" option if desired (headless = no visible browser)
4. Click "Start Test"
5. The system will execute the tests and display results in real-time

**4.4 Test Results Analysis**

After test execution:

1. View the test results in the visualization section
2. See statistics on passed, failed, and blocked tests
3. For failed tests, receive AI-generated suggestions on how to fix the issues
4. Use the analysis to improve your form implementation

**4.5 Report Generation**

Generate detailed test reports:

1. After test execution, click "Generate Report"
2. The system will compile results into a comprehensive report
3. View or download the report for documentation or sharing

**5. Working with the AI Assistant**

The AI Testing Assistant can help with:

* Generating test cases
* Analyzing test failures
* Providing suggestions for improvements
* Answering questions about testing best practices

To use the AI Assistant:

1. Type your request in the chat input field
2. The AI will respond with relevant information or actions
3. Follow the AI's suggestions to improve your testing workflow

Sample prompts:

* "Generate test cases for the Student Enrollment form"
* "Create a Playwright test script for login validation"
* "Analyze why the email validation test is failing"
* "How can I improve form accessibility testing?"

**6. Advanced Features**

**6.1 Custom Test Script Development**

For advanced users who need specialized tests:

1. Request a custom Playwright script from the AI
2. Modify the script as needed
3. Execute the custom script against your form

**6.2 Test Failure Analysis**

When tests fail:

1. The system analyzes the failure pattern
2. Categorizes issues by severity (high, medium, low)
3. Provides specific suggestions to fix each issue
4. Explains the reasoning behind each suggestion

**6.3 Integration with Form Builder**

The Testing Engine integrates with the Simplita Form Builder:

1. Forms created in the Form Builder are automatically available for testing
2. Test results can inform form improvements
3. Iterative testing helps ensure form quality

**7. Best Practices**

**7.1 Test Case Organization**

* Group related test cases by functionality
* Prioritize critical tests (form submission, required fields)
* Include edge cases and boundary testing

**7.2 Regular Testing**

* Run tests after any form changes
* Schedule periodic regression testing
* Monitor trends in test results over time

**7.3 Using Test Results**

* Address high-priority failures first
* Follow the AI's suggestions for fixing issues
* Document persistent problems for developer review

**8. Troubleshooting**

**8.1 Connection Issues**

If the Testing Engine shows connection errors:

1. Check your internet connection
2. Verify the Testing Engine status indicator
3. Click "Retry Connection" if needed

**8.2 Test Execution Failures**

If tests fail to execute:

1. Check if the form is accessible
2. Verify that test scripts are correctly formatted
3. Look for environmental issues (network, browser compatibility)

**8.3 Getting Help**

For additional assistance:

1. Ask the AI Assistant for help with specific issues
2. Check the documentation for updated information
3. Contact Simplita support for persistent problems

**9. Glossary**

* **Form Analysis**: Examination of form structure, fields, and validation rules
* **Test Case**: A specific scenario to validate form functionality
* **Headless Testing**: Running tests without displaying the browser interface
* **Test Results**: Outcomes of test execution (passed, failed, blocked)
* **Test Script**: Code that automates test execution, such as Playwright tests
* **AI Assistant**: The intelligent helper that guides testing activities
* **Test Analysis**: Evaluation of test results to identify issues and solutions