Artificial Intelligence-supported Software Test Automation in Development of SIMATIC Industrial Products

Student(s)

Emre Kaan Usta

Faculty Member(s)

Cemal Yılmaz

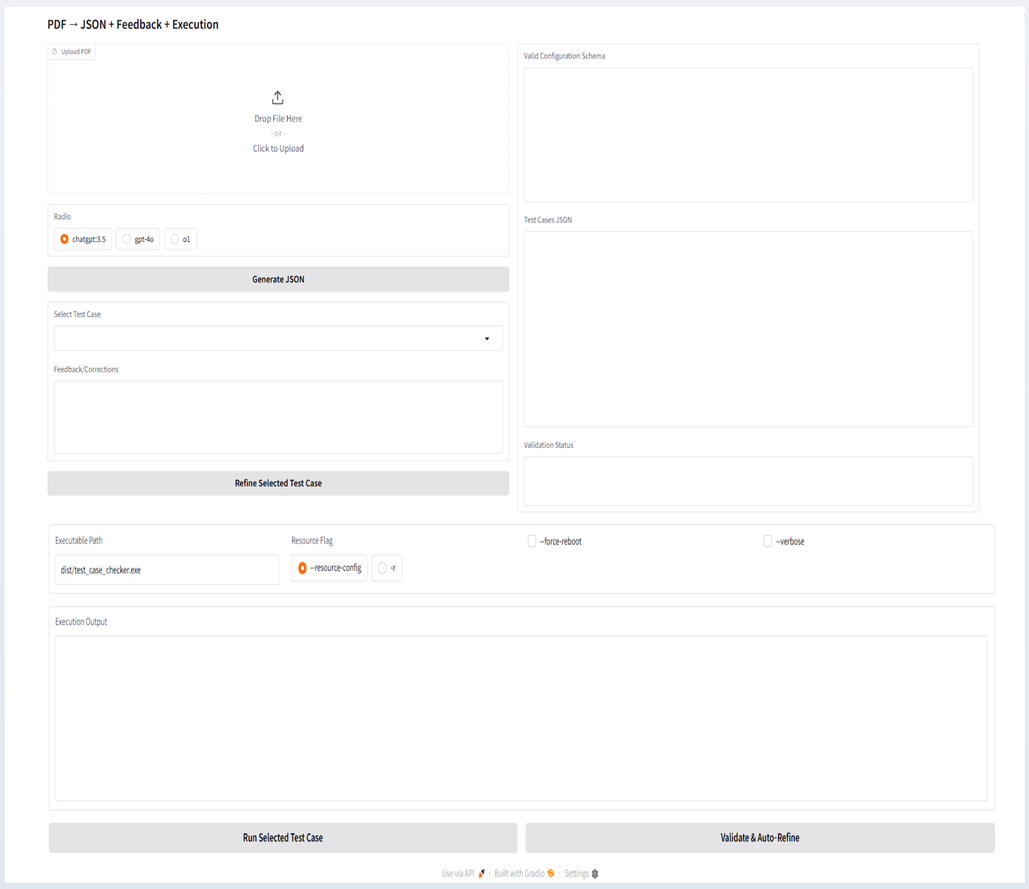
Company Advisor(s)

Haluk Şahin, Görkem İnci
Tutku Söker

Sabancı Üniversitesi
FACULTY OF ENGINEERING AND NATURAL SCIENCES

ABSTRACT

Manual authoring of JSON test cases slows down Siemens SIMATIC firmware validation, consuming up to 40% of engineers' time.
This project delivers an automated pipeline that converts Siemens specification PDFs into validated JSON test suites, including systematically generated negative tests. A Gradio-based interface allows instant validation, execution, and auto-refinement.
The prototype achieves a **12× speed-up** over manual authoring while maintaining **93% schema conformance**, demonstrating the potential of AI-assisted automation for safety-critical industrial software.



Methodology

PDF Parsing: Extracted specification text using PyPDF2 with Unicode-safe pre-processing.

LLM-Driven Generation: Prompted GPT-4-o to produce schema-valid JSON plus 15 negative test variants.

Validation: jsonschema + pydantic checks ensure strict conformance to Siemens' Test Configuration Schema v1.4.

Interactive UI: Gradio interface supports upload → JSON inspection → test execution → auto-refinement.

Closed-Loop Refinement: Failed test cases are re-submitted to GPT-4-o until convergence (max 3 iterations).

OBJECTIVES

•Automate conversion of Siemens SIMATIC specification PDFs into JSON test suites.

•Provide schema validation and negative test generation.

•Enable one-click execution and auto-refinement.

•Gather weekly Siemens feedback and refine UI/UX.

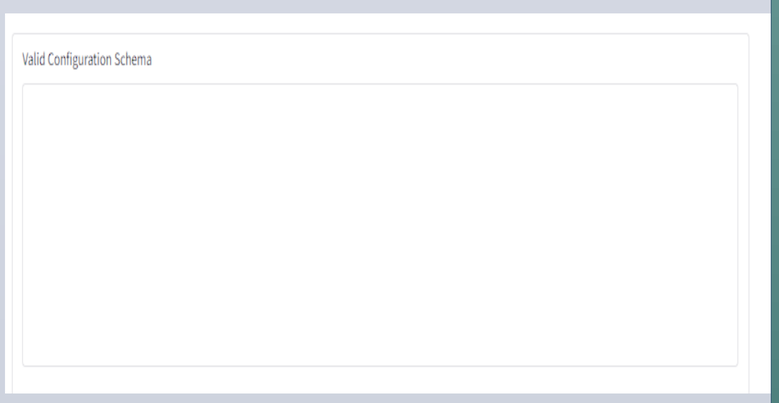
PROJECT DETAILS

Appendix 1: AI Service Selection:

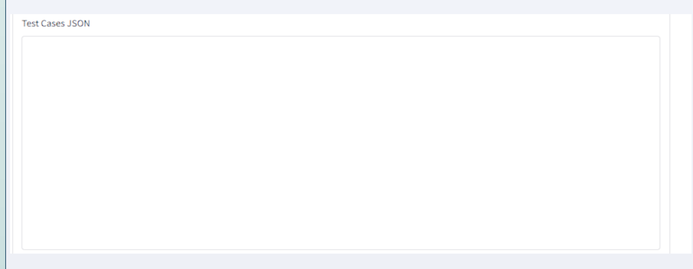
This UI section lets users pick the LLM for test case generation. Users can choose between mistral-7b-instruct, gpt-4, and a local model. This provides flexibility, allowing users to select a model based on cost or performance needs

Appendix 2: Valid Configuration Schema:

This panel shows a valid configuration schema derived directly from the uploaded PDF document. It helps users understand what a correctly formatted JSON output looks like. This is essential for user inspection and ensures the primary output is clear and verifiable.



PROJECT DETAILS II

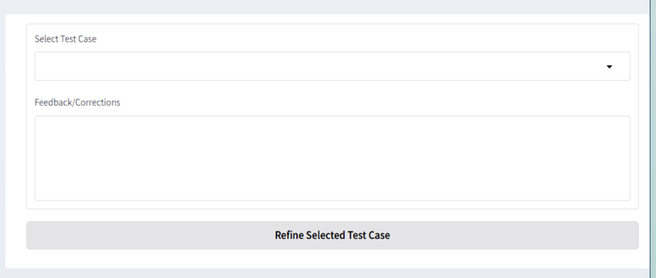


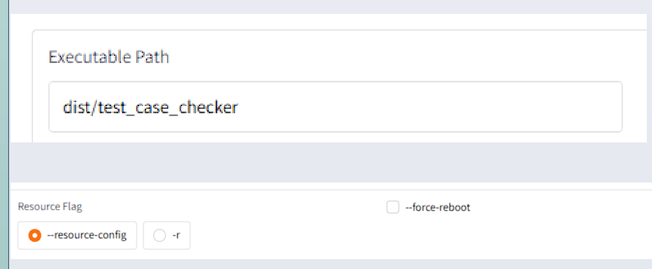
Appendix 3: Test Cases JSON:

This area displays all the generated test cases. The system creates these test cases as equivalence classes, which can be seen along with their names, expected outcomes, and the JSON data. This ensures a comprehensive set of test scenarios is created for thorough validation.

Appendix 4: Feedback and Refinement:

This feature enables users to manually refine test cases. Users can select a specific test case and provide feedback or corrections in a text box. The system then uses this feedback to refine the selected test cases, which are then shown again in the "Test Cases JSON" section.

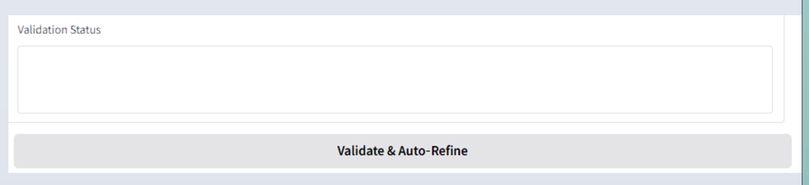




Appendix 5: Execution Controls:

This section allows users to specify the executable path for the Siemens test harness. It also provides options for adding command-line parameters, or "resource flags," directly from the UI. This functionality is key for one-click execution of the generated test cases.

Appendix 6: Automated Validation and Refinement:



This option allows the system to automatically compare expected outcomes with actual test results.
If there is a mismatch, the system automatically corrects its own output without the need for manual feedback. This closed-loop process is a key part of the auto-refinement workflow.

CONCLUSIONS

This project has successfully created a comprehensive, end-to-end toolchain that automates the generation of test cases for Siemens' SIMATIC firmware. The tool takes a specification PDF and converts it into a valid JSON test suite, which can then be executed using Siemens' existing test harness.

The solution was a collaborative effort with Siemens engineers, who provided weekly feedback to ensure the final product directly addresses their real-world needs and pain points. All core functionalities—including automated extraction, validation, execution, and an interactive correction loop—are fully operational and have been demonstrated with live documents.
By automating the most repetitive and time-consuming aspects of test-case authoring, this tool provides a solid foundation for faster and more reliable firmware validation. It reduces manual effort, maintains product safety by validating output against the official Siemens schema, and accelerates the feedback loop through an interactive user interface.

Objective	Current Status
Automate PDF → JSON generation	Implemented and demonstrated live in weekly calls
Provide schema validation	Default configuration and all negative tests checked against Siemens schema
Enable one-click execution	Run Selected Test Case button executes through test_case_checker
Gather continuous Siemens feedback	Weekly meetings held; requests tracked and fed into backlog
Expose adjustable equivalence-class count	Backend complete; front-end slider pending

REFERENCES

REFERENCES
[1] OpenAI. GPT 4o Model Card, 2025.
[2] Pedregosa F. et al. "Scikit learn: Machine Learning in Python." JMLR, 2011.
[3] Siemens AG. SIMATIC Test Configuration Schema v1.4, internal doc, 2024.