Enhancing OpenAPI Test Generation through Retrieval-Augmented Generation: A Case Study in Enterprise Hotel Software

Leon Kogler

May 6, 2025

1 Introduction

In recent years, the number of published studies on the testing of RESTful APIs has increased exponentially. This surge can be attributed to their extensive industrial application and potential to influence industrial practices significantly [1]. Despite the commendable performance of current tools in generating API Tests from OpenAPI Specifications (OAS) [2, 3], they often encounter challenges in interpreting the human-readable part of the specification [4, 5]. Kim et al. [5] analyzed 10 real-world REST APIs and identified more than 330 parameters and operation rules in the natural language part of the specification. The application of Large Language Models (LLMs) has gained prominence due to their enhanced reasoning capabilities in understanding complex problems. Recent studies have shown that using LLMs for automated API test generation can yield promising results [6, 7, 8]; however, existing tools rely only on the use of the OpenAPI specification and do not consider additional company-specific data (e.g., source code and internal/external documentation). Furthermore, these tools utilize a single Large Language Model (LLM) and lack comparative analyzes of their performance in API test generation.

This research aims to evaluate various LLMs in generating functional API Tests in the form of Postman collections [9] that can be executed using the Postman CLI (Newman), due to its integration into CI/CD pipelines. Furthermore, we investigate how Retrieval-Augmented Generation (RAG) [10] can improve the quality of generated tests by incorporating and assessing company-specific knowledge. To highlight both the limitations and opportunities of this research in real-world industrial environments, the evaluation will be conducted using the CASABLANCA Hotelsoftware system. CASABLANCA Hotelsoftware is an enterprise-grade Property Management System (PMS) characterized by a complex microservice architecture.

2 Research Questions

The main research questions of this work are as follows:

- **RQ1:** What criteria can be used to evaluate and compare the quality of REST API test generation tools?
- **RQ2:** How do different Large Language Models (LLMs) impact the quality of test generation for OpenAPI specifications?
- **RQ3:** What is the impact of utilizing RAG with additional data sources, such as code repositories and internal/external documentation, on the effectiveness of test case generation?
- **RQ4:** What limitations and potential opportunities does this approach present for the CASABLANCA use case?

3 State of the Art

A literature review shows that current research in the field of OpenAPI test generation using LLMs is still in its early stages. Some studies investigate the use of LLMs for test generation, but comprehensive analyses of the application of RAG in this context are lacking. Key sources include:

- *Retrieval-Augmented Test Generation: How Far Are We?* J. Shin et al. [11] investigates the efficacy of RAG-based LLMs in unit test generation. This study explores the impact of different sources of RAGs' knowledge bases, such as API documentation, GitHub issues, and StackOverflow Q&As, on unit test generation. The findings suggest that while RAG does not improve the syntactical or dynamic correctness of unit test cases, it can significantly enhance line coverage by providing unique states of programs that cannot be generated directly by LLMs. This may indicate that RAG is also beneficial for API test generation.

- *APITestGenie*, developed by A. Pereira et al. [6], represents a new approach that harnesses the context awareness and natural language processing capabilities of large language models (LLMs) to generate executable scripts for testing REST APIs. This methodology incorporates the Retrieval-Augmented Generation (RAG) technique to effectively handle OpenAPI specifications that exceed the context window limitations of LLMs. Notably, APITestGenie exclusively utilizes information derived from the OpenAPI specification and a user-provided description of the use case under test.

- *AutoRestTest*, published by T. Stennett et al. [7] in January 2025, introduces a multi-agent approach for REST API testing that integrates Multi-Agent Reinforcement Learning (MARL), a Semantic Property Dependency Graph (SPDG), and LLMs. AutoRestTest optimizes API exploration by treating REST API testing as a separable problem, where five specialized agents—operation, parameter, value, dependency, and header agents—collaborate. Similar to APITestGenie, AutoRestTest does not rely on additional company-specific or service-specific data.

4 Approach

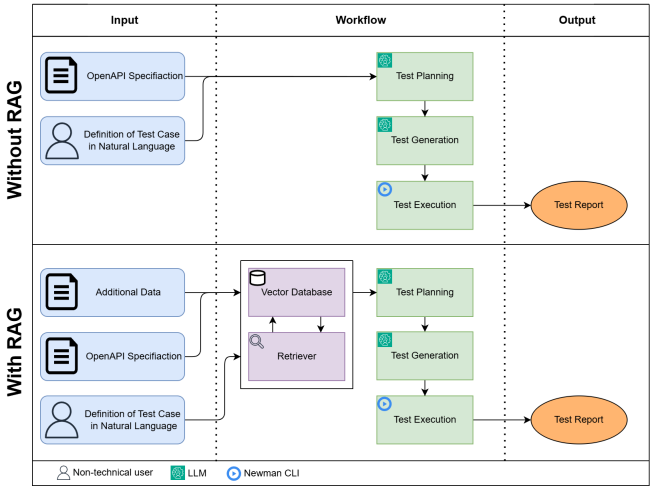

Figure 1: Workflow diagram of automated test generation using LLMs with and without RAG

RQ1: We conduct a literature review to define key evaluation criteria, enabling performance comparisons between our approach (with and without RAG) and existing tools.

RQ2: A baseline API testing tool without RAG, as shown in Figure 1, will be implemented utilizing LangChain [12]. This framework facilitates the integration of LLMs into applications and the construction of agentic workflows. To generate a test case, the tool will use a multi-step prompting workflow to create a test plan, generate the actual test cases and run the tests. To evaluate the performance of the tool, a test set will be created based on the criteria defined in RQ1.

RQ3: The generation of tests is enhanced by incorporating Retrieval-Augmented Generation (RAG) with a vector database, as shown in Figure 1, enabling the use of contextual data (e.g. source code, documentation) to improve test plans and cases. Performance is compared to the baseline tool of RQ2 using the same test set.

RQ4: To explore the limitations and potential opportunities of the proposed approach, an evaluation is conducted within the CASABLANCA Hotelsoftware use case. The evaluation considers factors such as cost, scalability, integration challenges, and operational boundaries, highlighting both constraints and areas where the tool demonstrates added value.

5 Work Plan

The total duration of the thesis is four months. A preliminary timeline for the execution of the work is shown in Figure 2.

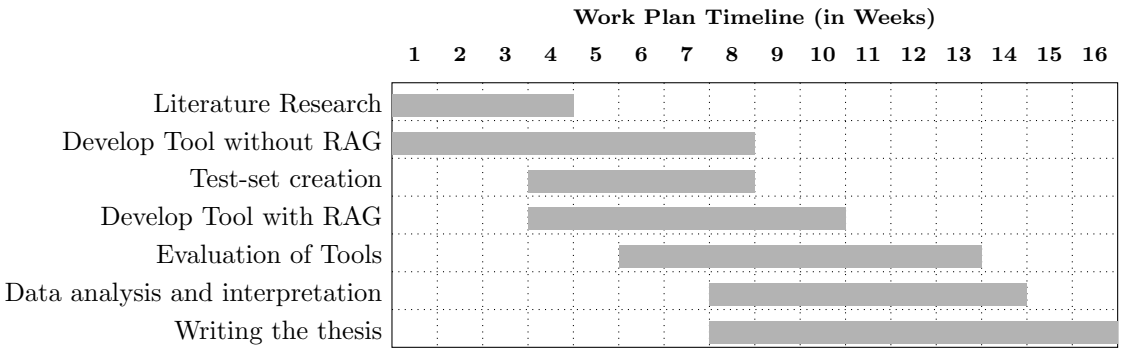

Figure 2: Thesis work plan timeline with tasks and their respective durations

References

[1] A. Golmohammadi, M. Zhang, and A. Arcuri, "Testing RESTful APIs: A Survey," 2022. [Online]. Available: https://arxiv.org/abs/2212.14604

[2] M. Kim, Q. Xin, S. Sinha, and A. Orso, "Automated test generation for REST APIs: no time to rest yet," in *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2022. New York, NY, USA: Association for Computing Machinery, 2022, p. 289–301. [Online]. Available: https://doi.org/10.1145/3533767.3534401

[3] OpenAPI, "OpenAPI standard," 2025. [Online]. Available: https://www.openapis.org

[4] M. Kim, T. Stennett, D. Shah, S. Sinha, and A. Orso, "Leveraging Large Language Models to Improve REST API Testing," 2024. [Online]. Available: https://arxiv.org/abs/2312.00894

[5] M. Kim, D. Corradini, S. Sinha, A. Orso, M. Pasqua, R. Tzoref-Brill, and M. Ceccato, "Enhancing REST API Testing with NLP Techniques," in *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2023. New York, NY, USA: Association for Computing Machinery, 2023, p. 1232–1243. [Online]. Available: https://doi.org/10.1145/3597926.3598131

[6] A. Pereira, B. Lima, and J. P. Faria, "APITestGenie: Automated API Test Generation through Generative AI," 2024. [Online]. Available: https://arxiv.org/abs/2409.03838

[7] T. Stennett, M. Kim, S. Sinha, and A. Orso, "Autoresttest: A tool for automated rest api testing using llms and marl," 2025. [Online]. Available: https://arxiv.org/abs/2501.08600

[8] M. Kim, S. Sinha, and A. Orso, "LlamaRestTest: Effective REST API Testing with Small Language Models," 2025. [Online]. Available: https://arxiv.org/abs/2501.08598

[9] Postman, "Postman Collections," 2025. [Online]. Available: https://www.postman.com/collection/

[10] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. tau Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," 2021. [Online]. Available: https://arxiv.org/abs/2005.11401

[11] J. Shin, R. Aleithan, H. Hemmati, and S. Wang, "Retrieval-Augmented Test Generation: How Far Are We?" 2024. [Online]. Available: https://arxiv.org/abs/2409.12682

[12] H. Chase and contributors, "Langchain: Building applications with llms through composability," https://python.langchain.com/, 2025, accessed: 2025-04-14.