SSRG International Journal of Computer Science and Engineering
ISSN: 2348–8387 / https://doi.org/10.14445/23488387/IJCSE-V12I18P104

Volume 12 Issue 8, 27-30, August 2025
© 2025 Seventh Sense Research Group®

Original Article

Importance of Structured Prompt Engineering to Generate Effective Test Cases

Nagmani Lnu

Director of Quality Engineer at a FinTech Company, San Antonio, TX, USA.

Corresponding Author : nagmanijobs@gmail.com

Received: 12 June 2025 Revised: 19 July 2025 Accepted: 08 August 2025 Published: 29 August 2025

Abstract - *As demand for Artificial Intelligence (AI) grows, best practices for using different AI tools are becoming critical. To scale AI implementation enterprise-wide, prompt engineering has become vital in developing AI-enabled applications and utilizing them to complete day-to-day engineering tasks. One such task is creating functional test cases. Software testing is a crucial stage in the Software Development Life Cycle and consumes a significant portion of the overall development timeline. This study was conducted while evaluating a test management tool that utilizes AI to generate test cases for UI (User Interface) and API (Application Programming Interface) applications. While generative AI is the future, using AI without proper guardrails and prompt standards can become counterproductive. This study demonstrates this by using multiple real-time industry use cases.*

Keywords - *Artificial Intelligence, Functional Test Case, Hallucination, Prompt Engineering, Software Testing.*

1. Introduction

Testing consumes a significant portion of the software development life cycle (SDLC). Depending on the project, it accounts for between 30% and 50% of the development effort [1]. The test life cycle includes two major phases: Test Planning and Test Execution. While there are many tools on the market to automate and execute test cases manually or automatically, few, if any, tools are available to create test cases. As a result, creating test cases requires a significant amount of effort in the overall testing life cycle. Artificial intelligence can become a valuable tool in this area, which is why around 68% of organizations in the Capgemini 2024 Quality Report mention plans to integrate AI into their tech stack [2].

As many organizations across industries are on the journey of integrating Artificial Intelligence (AI) into different phases of the SDLC to bring consistency and automation, prompt engineering is becoming critical. Therefore, in this study, the task was to evaluate various commercial tools available in the market, along with some popular LLMs such as Copilot and Windsurf. This study demonstrates that artificial intelligence, combined with effective prompt engineering, can increase the acceptability of AI-generated test cases by up to 100% and reduce test planning effort by up to 90%. This paper presents a comprehensive and practical use case and a prompt approach that can be applied across all industries.

2. Related Work

Prompt engineering is a relatively new concept that gained momentum in 2023. There are many papers coming around it. In 2025, Reangpusri published a paper exploring prompt engineering for Test Driven Development [4].

In 2024, L. Naimi, E. M. Bouziane, M. Manouch, and A. Jakime published a test case generation using LLM and prompt engineering [5]. No paper has focused on the effectiveness of the test cases and related measurements in this regard. This paper differentiates in this regard from all other research.

3. About Application and Testing Strategy

The application under test was a payment processing application that users can access to make payments for their loans or credit cards. Figure 1 below explains the high-level page flow of the application. Like other payment systems, this application accepts both credit card and ACH payments. Users have the option to make payments with their registered login or as guests.

When making payments, users must navigate to the relevant pages to select their account, enter payment details, and complete the verification and confirmation process. While this seems like a simple application, the effort required to create test cases varies depending on the situation.

- Situation 1- This could be an existing application. The team may already have existing test cases and wants to add or enhance them based on new features being added as part of the sprint.
- Situation 2- The application is in production, but comprehensive test cases are missing. The team is willing to put some effort into creating the regression tests that will be automated and executed along with the build and release process.
- Situation 3- Application is in the design phase and has an existing Figma design or flow diagram available to create test cases

All three situations were simulated during this study, and it was found that the quality of test cases and acceptance rate were very high when the AI was used with proper engineering, rather than simply passing the requirement as input.

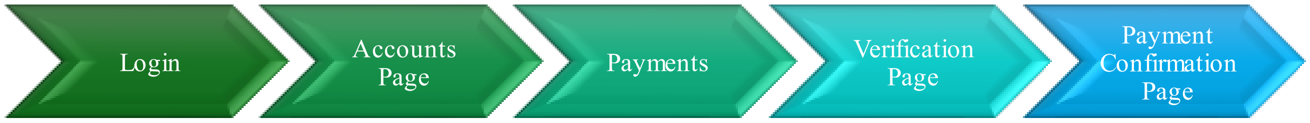

Fig. 1 Application flow

4. About Prompt Engineering

According to the Prompt Engineering Guide (https://www.promptingguide.ai/), researchers employ proper prompt engineering to enhance the LLM [3]. This is true not only for the developer who creates an AI application, but also for the user who interacts with an LLM to generate content.

The Prompt Engineering guide suggests many different prompt engineering techniques to interact with LL.M. A few of those are listed below.

- Zero-shot or Few-shot- In which the user tries to get the answer by providing zero to very few input contexts
- Chain of Thought - Where the user provides a chain of actions to perform a complex task
- Meta Prompting - Focus more on the structure and syntax of the prompt to generate content
- Graph prompt- To provide input on a graphical document (such as Figma, screenshot, or design document) in a standard form

Many other prompt engineering techniques can be found on the Prompt Engineering Guide website. These techniques are continually evolving, as new terms emerge in the industry.

5. Study Approach

The purpose of this study was to develop an enterprise-wide, scalable process to generate the maximum number of valid test cases, thereby minimizing review effort. A key metric, approval percentage, was considered to measure the success, which is equivalent to the total number of accepted test cases divided by the total number of generated test cases.

$$Approval\ percentage = \frac{Total\ Accepted\ TestCases}{Total\ Generated\ cases} * 100 \quad (1)$$

5.1. Approach 1- Experiment 1- Zero Shot Prompt - Requirement document

The experiment began by uploading a requirement document containing a detailed description of the application's functionality to the tool. The task was to generate end-to-end test cases without any prompts. AI tools generated 89 test cases in total; however, QE could only approve 19 of those, giving a very poor approval percentage of 23%

5.2. Approach 1- Experiment 2- Zero Shot Prompt – Detailed Requirement Document

This time, a different requirement document containing more details on application functionality was uploaded to generate test cases without any standard prompt. This time, the tool generated 51 test cases, but QE could accept only 6.

Again, the approval percentage was very poor at 11%. In both approaches, the quality of test cases was very poor, with missing test steps or invalid test cases that were completely out of context.

5.3. Approach2-Experiment 1 - Structured Prompt-Detailed Requirement Document

After the zero-prompt approach, the experiment was repeated with the following structured prompt.

5.3.1. Prompt

"Task: Generate comprehensive test cases for a specified use case flow

Context: As a software QA analyst, they need to organize the test cases by functional groups, ensuring broad coverage without redundancy. Each test case should be end-to-end, covering every step from beginning to end.

Tone: Professional"

Results were very promising. This time, the tool generated only 32 test cases, and out of those, 17 test cases were accepted, representing a significant jump in the acceptance rate from 11% to 53%. Mentioning Persona, as the QA analyst, with a descriptive prompt and a clear indication of the test case type, played a key role here.

5.4. Approach2-Experiment 2- With Structured Prompting – Requirement document passed as a table format

This time, the document was modified to a tabular form, presenting high-level scenarios. The task was to pass a structured prompt, as shown below, to generate detailed end-to-end test cases, which is more time-consuming. This is a very common situation where the QE team would like to design high-level test cases, but takes AI assistance to detail them out.

5.4.1. Prompt

"Task: Generate comprehensive test cases for each high-level test flow mentioned in the document
Context: as a Software QA analyst for each specified use case in the given document. Ensuring broad coverage without redundancy. Each test case should be end-to-end, covering every step from start to finish, and it should have a single standard flow within each test case. The test suite should have positive, negative, and all possible edge cases. Test cases should be based on both UI and functionality.
Tone: Professional"

Results were stunning, with the QE team accepting 100% of steps with very minimal modification.

5.5. Approach2-Experiment 3- Structured Prompting – Requirement document passed as screenshots flow

This time, the document was modified to include screenshots of the application flow, enabling the generation of end-to-end test cases. Once again, this is another very common use case in software engineering, where a team would like to use AI to generate test cases based on the application flow, such as a Figma design document or user interface screenshots.

5.5.1. Prompt

"Task: Generate comprehensive test cases for the flows depicted in the given document
Context: A Software QA analyst is responsible for each flow of the application mentioned in the given document. Ensuring broad coverage without redundancy. Each test case should be end-to-end, covering every step from start to finish, and it should have a single standard flow within each test case. The test suite should have positive, negative, and all possible edge cases. Test cases should be based on application functionality.
Tone: Professional"

5.5.2. Additional Prompt

An additional prompt was also included in each screenshot image to provide context for the AI. For example, the following additional context was added to the login page. "User has the option to skip login and go as a Guest to make payments. However, this option will be disabled for the user who does not have this set up enabled."

An additional prompt helped the AI agent to create three sets of test cases to cover all the required flows.

- Submit payment with a registered user
- Submit payment with guest permission when enabled.
- Guest payment options are not available if it was not configured for the bank.

Results were stunning, with the QE team accepting 96% of test cases with very minimal modification.

5.6. Approach2-Experiment 4- With Structured Prompting – Requirement document was passed as Swagger documentation

After UI test cases, the attempt was changed to generate API test cases. A Swagger documentation containing all API definitions was uploaded to generate end-to-end API test cases for the application.

5.6.1. Prompt

"Task: Generate comprehensive end-to-end API test cases for the given Swagger documentation
Context: A Software QA analyst is responsible for the API definition mentioned in the document. Ensuring broad coverage without redundancy. Each test case should be end-to-end, covering every step from beginning to end. The test suite should have positive, negative, and all possible edge cases. Test cases should be based on application functionality."

Results were stunning, with the QE team accepting 94% of test cases with very minimal modification.

6. Conclusion

This study proves that teams with a good understanding of the system and requirements can generate highly effective test cases, with an acceptance rate of 94 to 100% for UI and APIs, by providing a structured prompt (as depicted in Table 1 below). In the absence of a structured prompt, Quality Engineers will spend an enormous amount of time removing unwanted test cases and will not utilize Artificial Intelligence technology to its fullest capability.

Prompts can be provided at the beginning to trigger the task process for AI, and if needed, within documents such as Figma documents or Swagger documentation, to help AI generate a quality test case. A structured prompt should clearly include the task and context to avoid any hallucinations. For example, if a user wants to generate test cases for a specific requirement (such as a small requirement or a defect), rather than for end-to-end test cases, they should specify this clearly in the prompt.

This study helps companies create an enterprise-wide Artificial Intelligence prompt engineering practice with standards and guardrails.

Table 1. Results summary

Approach	Exercise	Prompt Type	Use Case	Approval Percentage
Approach1	1	Zero Shot	End-to-End UI Test Case	23%
Approach1	2	Zero Shot	End-to-End UI Test Case	11%
Approach2	1	Structured	End-to-End UI Test Case	53%
Approach2	2	Structured	End-to-End UI Test Steps	100%
Approach2	3	Structured	End-to-End Test Case from Figma	96%
Approach2	4	Structured	End-to-End API Test Case from Swagger	94%

References

[1] Time Estimation for Software Testing, Devmio - Software Know-How, 2016. [Online]. Available: devm.io/testing/time-estimation-for-software-testing-128078.

[2] Sahelichakraborty, World Quality Report 2024 Shows 68% of Organizations Now Utilizing Gen AI to Advance Quality Engineering, Capgemini USA, 2024. [Online]. Available: www.capgemini.com/us-en/news/press-releases/world-quality-report-2024-shows-68-of-organizations-now-utilizing-gen-ai-to-advance-quality-engineering/

[3] Prompting Techniques – Nextra, Prompt Engineering Guide. [Online]. Available: www.promptingguide.ai/techniques

[4] Theodore Reangpusri, and Cassandra Flur, "Test-Driven Development: Exploring Prompt Engineering," *Digital Scientific Archive*, 2025. [Google Scholar] [Publisher Link]

[5] Lahbib Naimi et al., "A New Approach For Automatic Test Case Generation From Use Case Diagram Using LLMS and Prompt Engineering," *International Conference on Circuit, Systems and Communication*, Fes, Morocco, pp. 1-5, 2024. [CrossRef] [Google Scholar] [Publisher Link]