Test Case Creation Approach using LLM for Android System

Bruno Cerdeira[1], **Hermann Hernani**[1], **Luan Marques**[1], **Luiz Souza**[1], **Silvia Ascate**[1]
Eliane Collins[1], **André Carvalho**[2]

[1]Instituto de Desenvolvimento Tecnológico – (INDT)
Av. José Moacir Teberga de Toledo, 1520 - Planalto, Manaus - AM, 69044-235

[2]ICOMP – Instituto de Computação
Universidade Federal do Amazonas (UFAM), Manaus – AM, Brazil.

bruno.cerdeira@indt.org.br, hermann.oliveira@indt.org.br
luan.marques@indt.org.br, luiz.souza@indt.org.br, silvia@indt.org.br
eliane.collins@indt.org.br, andre@icomp.ufam.edu.br

Abstract. *This research project presents a method to generate functional test cases for new Android features using the Large Language Model (LLM) LLama. The objective is to reduce manual effort and avoid process bottlenecks, benefiting software testing teams and developers. The approach uses LLM-generated prompts based on Android feature specifications for automated test case generation. A feasibility study evaluated 57 test cases, revealing that 19.3% were well-written, 63.2% had acceptable quality, and 26.3% had medium complexity. The research was conducted under contract with the Institute of Technological Development (INDT) and its funded by the R&D program of EMBRAPII for Motorola LTDA/Flextronic Ltda.*

1. Context

As the mobile system market is very competitive, there is a constant need for effective strategies to ensure the delivery of high-quality applications. Software testing is fundamental in the development process that aims to evaluate and improve the quality of products. Mobile systems have some characteristics and limitations such as the battery life, amount of input data, and different inputs. This context increases the complexity of test case maintenance. Furthermore, generating test scripts requires constant monitoring and updates for new devices, OS versions, resolution variations, and Google adopts a rapid Android release cycle with frequent updates and new features.

New techniques have been used to solve these gaps, such as Artificial Intelligence, which has contributed significantly to the generation of tests. LLMs provide language understanding abilities and adaptability across various fields, including software testing. It can be used to create functional test cases for specific contexts Lima Jr et al. (2024).

LLMs have been successfully implemented in various testing tasks. Liu et al. (2023) propose GPTDroid, a system that uses LLMs to generate testing scripts to provide Graphical user interface (GUI) page information to the LLM for mobile apps and execute them. In Lima Jr et al. (2024), a case study investigates the integration of LLMs in creating test cases. The authors generate test cases using a real software application and evaluate their effectiveness based on quality factors like Clarity, Correctness, Completeness, and Consistency.

2. Study Objectives

The aim of this research project is to evaluate the feasibility of incorporating the LLM Llama 2.0[1] (Llama2) model into the test case creation process in the mobile systems industry, without the need for additional training, based on the analysis of the quality and complexity of the generated tests. Quality was analyzed following criteria such as clarity, completeness, objectivity, and maintainability Lima Jr et al. (2024), while complexity was evaluated considering the test case configuration, test steps, number of components, and test case abstraction level Tran et al. (2021).

These are the contributions of this study: A process to create test cases from Llama 2.0 helping test professionals and students; evaluation of the test cases created using the quality and complexity metrics; and benefits professionals in the Android software testing market and companies in this business.

3. Methods and Technologies

INDT - Instituto de Desenvolvimento Tecnológico, provides solutions in the areas of Software, Hardware & Firmware, Communication and Networks, Advanced Manufacturing, Autonomous Vehicles and Robotics, Materials and Chemistry, and BioTech. The projects are carried out in collaboration with companies in these sectors, in which problems are identified and ways to solve them are developed through research and development.

In this context, in March 2024 the partner company had to re-think how it would handle with the advance of LLMs in the context of creating test cases for software engineering. Consequently, this research project initially aimed to prove the feasibility of using LLMs to create test cases in large-scale systems that met the criteria already mentioned in the previous section (quality and complexity). The feasibility study has the following process flow represented in Figure 1, with four stages, from describing the features to refining the final result after generating the test case using the Llama2.0 script.

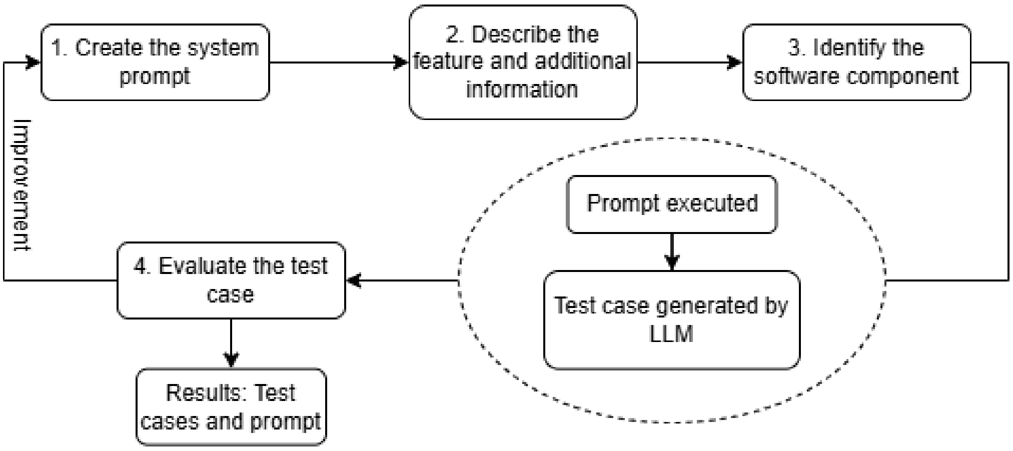
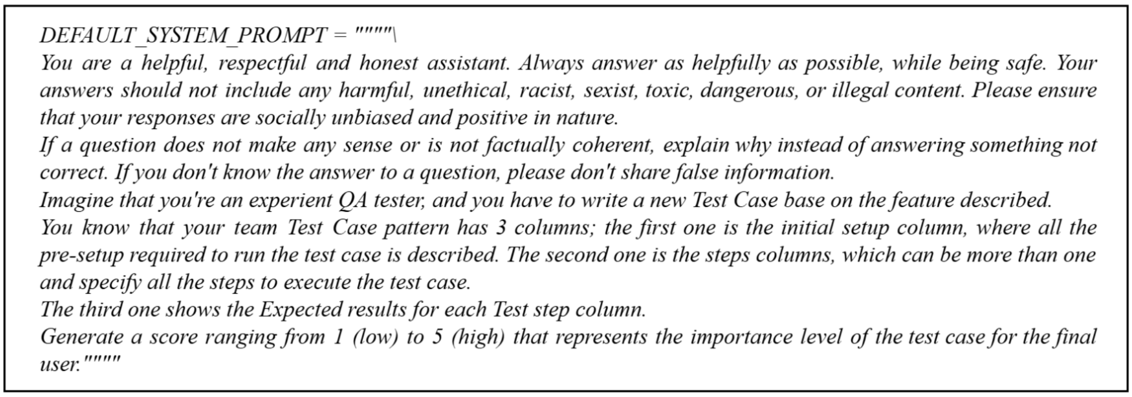

1. Create the system prompt
2. Describe the feature and additional information
3. Identify the software component
Improvement
Prompt executed
Test case generated by LLM
4. Evaluate the test case
Results: Test cases and prompt

Figure 1. The proposed solution

The system prompt was initially created using the Zero-shot prompting technique, providing a natural language description of the task without prior examples Jaremko et al. (2025). Guidelines for impersonation, response formatting, and evaluation criteria were

[1] Meta. 2024. Llama 2. https://llama.meta.com/

included to enhance the model's output. A detailed feature description (2) was then provided to help understand its functionality and objectives, facilitating the generation of test cases. Identifying the software components that interact with these features clarifies dependencies, enabling the LLM to create targeted test cases focused on specific interactions (3). The evaluation process (4) involved testers manually reviewing and executing the generated test cases to ensure they met quality and complexity standards.



```
DEFAULT_SYSTEM_PROMPT = """\
You are a helpful, respectful and honest assistant. Always answer as helpfully as possible, while being safe. Your
answers should not include any harmful, unethical, racist, sexist, toxic, dangerous, or illegal content. Please ensure
that your responses are socially unbiased and positive in nature.
If a question does not make any sense or is not factually coherent, explain why instead of answering something not
correct. If you don't know the answer to a question, please don't share false information.
Imagine that you're an experient QA tester, and you have to write a new Test Case base on the feature described.
You know that your team Test Case pattern has 3 columns; the first one is the initial setup column, where all the
pre-setup required to run the test case is described. The second one is the steps columns, which can be more than one
and specify all the steps to execute the test case.
The third one shows the Expected results for each Test step column.
Generate a score ranging from 1 (low) to 5 (high) that represents the importance level of the test case for the final
user."""
```

3.1. Feasibility Study

Writing quality was evaluated using the following factors: clarity, objectivity, completeness, and maintainability, each rated on a scale from 1 to 3 in which 1 represents poor, 2 is acceptable, and 3 is good. Clarity assessed the precision and comprehensibility of the test case descriptions. Objectivity focuses on the conciseness and relevance of the information provided. Completeness ensured that the test case included all necessary setup details and expected outcomes. Maintainability examined the ease with which the test case could be adapted to future changes.

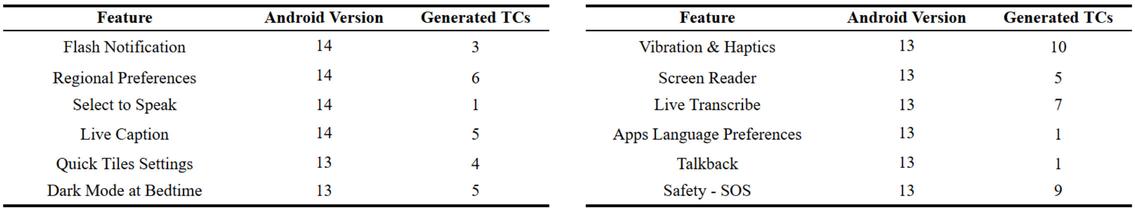
Test case complexity was rated from 1 to 3, considering setup, steps, components involved, and abstraction level. The Llama2.0 model was implemented in a Google Colab Python script using Android features. Four test analysts created the test cases, and two professionals evaluated their quality based on writing and complexity.

4. Results

Using the developed prompt, 57 test cases were generated based on the requirements outlined in Figure 2. Some test cases lacked expected results for each step, affecting completeness, while missing or inadequate initial setup data impacted complexity. A limitation of the method is the need for test analysts to map interacting features manually, and the limited number of evaluated Android features lacks statistical relevance.

The results show that 82.5% of the generated test cases had acceptable quality, with 63.2% partially acceptable due to issues with completeness and complexity, and 19.3% fully acceptable without modifications. The approach worked well for high-level descriptions, such as the "Select to Speak" feature, enabling the creation of comprehensible and effective test cases. Minimal modifications were needed, facilitating integration into the platform's overall test suite.

The LLM's non-persistent nature affected test consistency, making it challenging to generate accurate test cases and apply few-shot strategies effectively. Additionally, varying the number of steps in a test case proved difficult.



Feature	Android Version	Generated TCs
Flash Notification	14	3
Regional Preferences	14	6
Select to Speak	14	1
Live Caption	14	5
Quick Tiles Settings	13	4
Dark Mode at Bedtime	13	5

Feature	Android Version	Generated TCs
Vibration & Haptics	13	10
Screen Reader	13	5
Live Transcribe	13	7
Apps Language Preferences	13	1
Talkback	13	1
Safety - SOS	13	9

Figure 2. Google Android Features used in the feasibility study

Despite the good quality of test creation, there are opportunities for improvements to achieve completeness and complexity, especially when it comes to understanding how different features interact with each other. These improvements aim to benefit test professionals, students, and software testing teams and developers seeking to optimize Android feature validation by providing more comprehensive and effective test cases. In future works, the LLM prompts will be updated with information about the initial setup and the Android platform description for the model to infer the components interact with a specific feature.

Acknowledgment

This research was done during the year 2024 and was funded as provided for in arts. 21 and 22 of decree no. 10,521/2020, under the terms of Federal Law no. 8,387/1991, through agreement no. 003/2021, signed between INDT - Instituto de Desenvolvimento Tecnologico, Flextronics da Amazonia Ltda and Motorola Mobility Comercio de Produtos Eletrônicos Ltda.

References

Julia Jaremko, Dagmar Gromann, and Michael Wiegand. Revisiting implicitly abusive language detection: Evaluating llms in zero-shot and few-shot settings. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 3879–3898, 2025.

Roberto F Lima Jr, Luiz Fernando PB Presta, Lucca S Borborema, Vanderson N Silva, Marcio LM Dahia, and Anderson Santos. A case study on test case construction with large language models: Unveiling practical insights and challenges. In *Congresso Ibero-Americano em Engenharia de Software (CIbSE)*, pages 388–395. SBC, 2024.

Zhe Liu, Chunyang Chen, Junjie Wang, Mengzhuo Chen, Boyu Wu, Xing Che, Dandan Wang, and Qing Wang. Chatting with gpt-3 for zero-shot human-like mobile automated gui testing. *arXiv preprint arXiv:2305.09434*, 2023.

Huynh Khanh Vi Tran, Michael Unterkalmsteiner, Jürgen Börstler, and Nauman bin Ali. Assessing test artifact quality - a tertiary study. *Information and Software Technology*, 139:106620, 2021.