



## **REST-at: An LLM-Based Tool for Automating Traceability between Requirements and Test Cases**

Downloaded from: <https://research.chalmers.se>, 2026-10-13 18:31 UTC

Citation for the original published paper (version of record):

Quinstedt, N., Lindgren, B., Yurdakul, M. et al (2026). REST-at: An LLM-Based Tool for Automating Traceability between Requirements and Test Cases. Proceedings 2026 7th ACM IEEE International Conference on Automation of Software Test Ast 2026: 1-11.  
<http://dx.doi.org/10.1145/3793654.3793746>

N.B. When citing this work, cite the original published paper.

© 2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, or reuse of any copyrighted component of this work in other works.

# REST-at: An LLM-Based Tool for Automating Traceability between Requirements and Test Cases

Nicole Leon-Quinstedt

Chalmers University of Technology and the University of  
Gothenburg  
Gothenburg, Sweden  
nicole.quinstedt@gmail.com

Mert Yurdakul

Test Scouts Sweden AB  
Gothenburg, Sweden  
mert.yurdakul@testscouts.se

Bao Lindgren

Chalmers University of Technology and the University of  
Gothenburg  
Gothenburg, Sweden  
guslindgba@student.gu.se

Francisco Gomes de Oliveira Neto

Chalmers University of Technology and the University of  
Gothenburg  
Gothenburg, Sweden  
francisco.gomes@cse.gu.se

## Abstract

Large Language Models (LLMs) are increasingly used across Software Engineering tasks, and their ability to process natural language makes them particularly well suited for aligning requirements and testing artifacts. In this study, we investigate AI-supported traceability in software development through the creation and evaluation of REST-at, a tool that facilitates alignment between requirements engineering and software testing (REST) by automatically generating trace links between requirements and manual system-level tests. To evaluate its effectiveness, we compare five LLMs, including proprietary models such as GPT-4 and open-weight alternatives like Mixtral 8x7B and LLAMA 3 70B. The results indicate that open-weight models can achieve comparable or even superior levels of precision and recall relative to larger proprietary models. Among them, LLAMA 3 70B consistently provided robust and reliable link generation between requirements and tests. We also observe a high sensitivity to prompt design, with model performance varying according to prompt structure. Based on these insights, we provide practical recommendations for researchers and practitioners concerning hardware demands, governance and transparency, and the importance of standardized documentation when applying open-weight models to software engineering artifacts.

## CCS Concepts

• **Software and its engineering** → *Software design engineering*; **Software maintenance tools**; • **Computing methodologies** → **Natural language processing**.

## Keywords

Software Testing Traceability, Prompt Engineering, Large Language Models

## ACM Reference Format:

Nicole Leon-Quinstedt, Bao Lindgren, Mert Yurdakul, and Francisco Gomes de Oliveira Neto. 2026. REST-at: An LLM-Based Tool for Automating Traceability between Requirements and Test Cases. In *7th ACM/IEEE International Conference on Automation of Software Test (AST '26)*, April 13–14, 2026, Rio de Janeiro, Brazil. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3793654.3793746>

## 1 Introduction

Aligning Requirements Engineering (RE) with Software Testing (ST) can contribute to improving the effectiveness of software development. While RE focuses on capturing and defining what a system should achieve [33, 34], ST verifies whether the implemented software fulfills those requirements and complies with quality standards [3, 34]. This interdependence makes the alignment between RE and ST, hereafter referred to as *REST alignment*, a key element in ensuring software quality.<sup>1</sup> In practice, large projects often rely on traceability mechanisms to strengthen this alignment [3]. Established strategies include adopting traceability tools [19], integrating metrics, and promoting closer interaction between testers and requirements stakeholders [36]. Even with these measures, the process of manually connecting requirements and test cases remains labor-intensive and prone to human error [29].

Artificial Intelligence (AI) techniques, particularly Natural Language Processing (NLP), have opened new possibilities for automating REST alignment [12, 14]. In recent years, several studies have explored the use of Large Language Models (LLMs) to establish trace links between requirements and system-level tests [16, 29]. Some approaches build on encoder-only architectures such as BERT [11, 29], while others employ decoder-only models that not only identify missing trace links but also generate new artifacts, including test cases for uncovered requirements [16]. The quality of the prompt or prompt template used has been shown to significantly affect how well these models perform [11, 13, 22, 25].

Our paper presents REST-at, an AI-based tool that leverages decoder-only language models to automatically identify trace links between requirements and system-level test cases. We evaluate



This work is licensed under a Creative Commons Attribution 4.0 International License. *AST '26, Rio de Janeiro, Brazil*

© 2026 Copyright held by the owner/author(s).  
ACM ISBN 979-8-4007-2476-3/26/04  
<https://doi.org/10.1145/3793654.3793746>

<sup>1</sup>The acronym REST was defined in prior traceability research [33] to refer to the alignment between requirements and testing artifacts and should not be confused with REST (Representational State Transfer) used in web API design.

REST-at on plain-text artifacts, where requirements are short natural language descriptions and system-level tests. The tool was developed in collaboration with our industry partner, Test Scouts, a consultancy specializing in software testing. We further analyze how different language models and prompt engineering strategies affect trace link quality. Our research questions are:

**RQ1: To what extent do different large language model families impact the effectiveness of REST alignment?**

We evaluate REST alignment by identifying correct trace links between requirements and tests in two datasets: one from industry and one open-source. We compare five large language models spanning open- and closed-weight approaches, with varying parameter scales, and analyze how these characteristics affect trace link quality. Mistral has 7 billion (B) parameters [17], Mixtral has 46.7B parameters with 13B active, and LLAMA has 70B parameters. We also compare with the proprietary models GPT-3.5 (175B parameters [5]), GPT-3.5 Turbo and GPT-4's parameters from OpenAI.

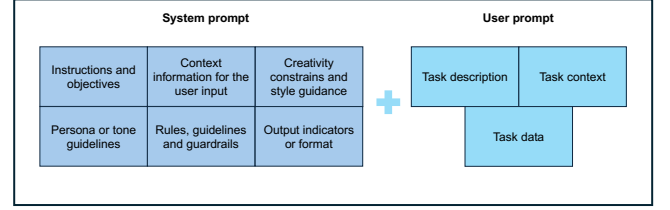
**RQ2: How do different prompt template components impact LLM performance in REST alignment?**

Prompt templates are divided into two components: the *user prompt*, which specifies the task, and the *system prompt*, which provides predefined instructions on how the LLM processes the user prompt [2, 38]. Using the same setup, we assess the effect of different system prompts and context structures inspired by prompt patterns [37] on REST-at.

Alongside the empirical comparison of models, we highlight practical challenges that emerged during the development of REST-at. These include execution costs and data governance issues related to proprietary models, as well as the inconsistent or incomplete documentation often found in open-weight alternatives. Note that our evaluation was conducted between May and July 2024. Extending the experiments to newer model versions is part of the planned follow-up work in collaboration with our industry partner, who has not yet approved additional experimental runs on their infrastructure.

Our recommendations focus on the practical aspects of deploying REST-at with open-weight models. We also describe lessons learned concerning software and hardware integration, security requirements, and the technical costs of running LLMs in local environments. In turn, our findings show that model size has no clear practical effect on REST alignment effectiveness, as precision, recall, and  $F_1$  scores remained high across models, with Mixtral and LLAMA 3 emerging as the most reliable for comprehensive alignment. We also observe that prompt structure strongly influences model behavior, with each LLM responding differently, and LLAMA achieving 100% average recall across most prompts on the industrial dataset.

Although the evaluation is based on previous model versions, our insights target principles and practices that remain relevant across future generations of models. Our contributions are:



**Figure 1: Representation of prompt components and their structure [2, 20, 26].**

- **REST-at**, an open-source tool that generates trace links between requirements and manual system-level tests using LLMs. Our repository also enables replication of our analysis and further extension by researchers and practitioners [1].
- A set of practical recommendations for simplifying the deployment and configuration of non-proprietary models in software engineering settings.
- A systematic comparison of multiple LLMs in the context of REST alignment. For example, Mixtral and LLAMA 3 demonstrated higher recall than proprietary GPT models when identifying trace links.

## 2 Background and Related Work

Here, we clarify the key concepts used in our paper. Specifically, we distinguish between open, closed, and open-weight models, as well as prompt template, prompt structure, or prompt engineering.

*Open-source* models refer to models whose source code and architecture are publicly accessible, enabling developers to understand, improve, and build upon existing models. On the other end of the spectrum, we find *closed-source*, where the source code, architecture, and datasets used for training are private [23, 35]. In between, we find open-weight models where the weights of the network are publicly available. *Open-weight* models allow developers to use and fine-tune the model but do not provide full transparency since the source code is unavailable [23, 35]. This distinction is important because it affects how much control and insight developers have over the model's functionality and behavior.

A *prompt* is an input, often text-based, sent to an LLM to generate an output response. The input is often text, but other formats (e.g., audio or image) are also acceptable. *Prompt engineering* is a technique to create or refine prompt templates, i.e., a structured prompt used with decoder-only LLMs such as GPT [13, 22]. Examples of prompt templates are: Few-Shot, Chain-of-Thought (CoT), Role-Playing, etc. *Prompt template* comprises the system and the user prompt (Figure 1) [32].

The *user prompt* should contain two important elements: the task, which specifies the expected result [32], and the context, which provides background information regarding the task and data on which an operation will be performed. In contrast, the system prompt provides specific instructions, guidelines, and context for the user prompt serving as the initial context and starting point for the model [2, 24, 38]. The *system prompt* in AI chatbots is often a persona description [32] together with guidelines and rules to avoid jailbreaking, which are usually hidden from the user. The system and user prompt are concatenated, guiding the model to a desired

output. However, the impact of each prompt component, as well as different prompting techniques, is an ongoing subject of research [11, 18]. Therefore, we aim to investigate the impact of such techniques on the models' ability to identify trace links between requirements and test artifacts.

## 2.1 AI Advances in REST Alignment

Many tools have been proposed to support REST alignment, particularly by applying ML techniques. Felderer et al. [14, 31] showed the effectiveness of utilizing NLP and unsupervised ML techniques to identify similar or duplicate test case specifications, optimizing manual system testing. Similarly, Qiu and Jin [25] compare the performance between two different LLMs, in text generation or classification tasks. The study emphasizes the potential of GPT-3.5 for driving advancements in AI across various fields.

The traceability tool created by Sawada et al., [29] automatically links software requirements with both automated and manual test cases with 75% accuracy, using a fine-tuned version of BERT. To support the maintainability of software projects, Dearstye proposes an approach for generating trace links and corresponding specification artifacts from existing project artifacts [11]. This approach leverages BERT models to identify feature relationships by clustering related artifacts. Their evaluation, however, was limited to two scenarios and primarily addressed safety-critical compliance contexts. In contrast, our work applies decoder-based LLMs and focuses exclusively on generating trace links for existing artifacts rather than producing new specification artifacts.

With a different approach, the traceability tool developed by Ivarsson and Setterström uses GPT-3.5 together with a combination of prompting techniques to provide feedback to testers and create a traceability report between requirements and system-level test cases. Their approach resulted in 86.3% recall, but with only 45.5% mean precision [16].

Those traceability tools above showcase how AI technologies can enhance the efficiency and effectiveness of quality assurance practices in the software industry, despite (i) the complex challenges of automating REST at system-level tests [14], and (ii) the nondeterminism and lack of explainability behind LLMs<sup>2</sup> [12]. Furthermore, using AI models through API providers introduces data and privacy risks [9, 12, 16].

Our study complements the literature by (i) providing more quantitative evidence of the accuracy and precision of using LLMs for REST alignment and (ii) revealing the trade-off between proprietary and open-weight models, particularly regarding aspects of security, costs, and governance of data when building REST alignment tools. Therefore, our findings can support practitioners in choosing a model that better fits their context.

## 3 AI-Driven Traceability with REST-at

REST-at aims to reduce the time and costs of validating requirements and tests by aiding testers in detecting trace links. We emphasize that REST-at is a supporting tool to improve REST alignment, not to replace human testers. The requirements for REST-at were

gathered through two main approaches: (i) investigating existing tools that use decoder-only LLMs for REST alignment [16], and (ii) holding meetings with our industry partner. A key requirement identified from both sources is that REST-at must (i) be flexible, (ii) support a variety of open-weight models, and (iii) enable communication with proprietary models via APIs (e.g., OpenAI). Figure 2 illustrates how REST-at works.

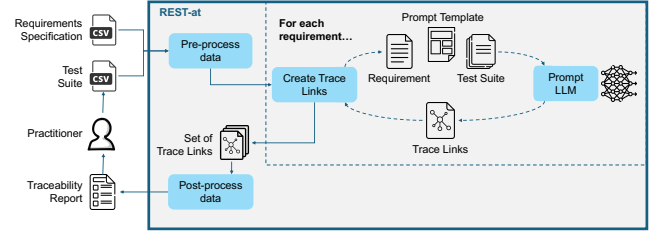


Figure 2: Overview of REST-at and its components.

The REST-at tool processes two types of input artifacts: a requirements specification and a corresponding test specification. These artifacts are provided as CSV files, after which REST-at automatically establishes mappings between them and produces a traceability report that connects each requirement to the relevant test cases. Our current evaluation focuses on system-level artifacts expressed in natural language (Section 4). Our future work aims to examine how the tool performs when linking requirements to source code artifacts, such as unit tests.

To generate the trace links, the tool interacts with the LLM by submitting one requirement at a time, along with the complete set of test cases. The model responds with the trace links that associate the given requirement with matching tests. Internally, the requirement and the test suite are merged into a structured prompt template that instructs the model to return its output as a table. The table includes the following columns: Requirement ID, Description, Tested, and Tested by. The “Tested” column specifies whether the requirement is covered, using “yes” or “no,” and the “Tested by” column lists the IDs of the corresponding test cases. Future versions of the tool aim to integrate RAG (Retrieval Augmented Generation) or agents to optimize access to the repository of requirements and tests, as well as context about the development project.

### 3.1 Design, Frameworks, and Infrastructure

REST-at (Figure 3) is built with Python and Streamlit, a web framework for data science and machine learning applications [27, 30]. Streamlit uses Python to create React<sup>3</sup> components under the hood, ensuring a modern and responsive user interface. Python is the primary language for working with AI models, which provides Streamlit applications seamless integration with popular AI libraries such as TensorFlow, PyTorch, or Transformers [30].

REST-at communicates with the LLM via requests, to allow for different ways to deploy an LLM. This allows REST-at to send requests to proprietary APIs or a service that hosts the LLMs, hence enabling flexibility to seamlessly switch between LLMs. A vast amount of vRAM was required to run the non-proprietary models

<sup>2</sup>Explainability refers to the ability of an AI system to provide clear explanations for its decisions or outputs, allowing the user to understand the reasoning process behind it [7]

<sup>3</sup><https://react.dev>

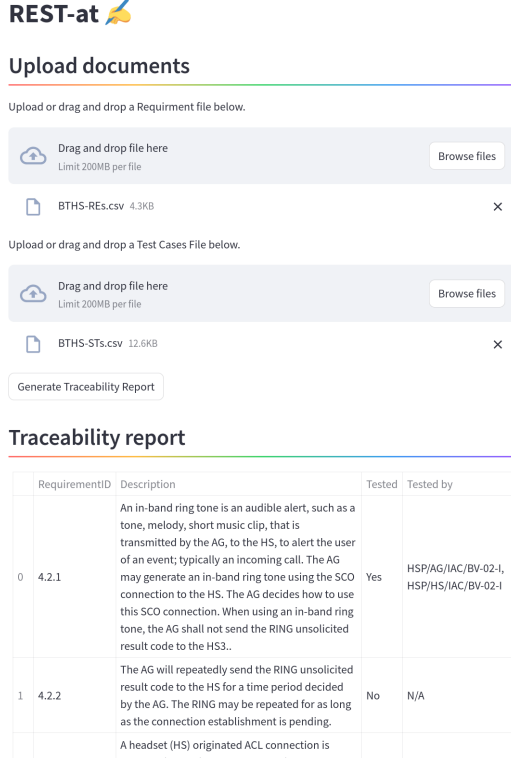


Figure 3: Main screen of REST-at.

locally (Mistral, Mixtral, and LLAMA). Mistral requires at least 16GB vRAM, Mixtral requires at least 100GB vRAM, and LLAMA 3 requires around 150GB vRAM.

To deploy and evaluate REST-at, we used C3SE (Chalmers e-Commons e-Infrastructure), a research infrastructure at Chalmers University of Technology in Sweden to minimize hardware costs. Our industry partner covered the costs of using the proprietary APIs. During development, we refined the content and format of the files and the prompt template to minimize both financial and computational resources when running the LLM. Currently, REST-at primarily uses JSON to exchange data for trace links and report creation. Multiple libraries are available to convert JSON files to other formats, such as XML or CSV, facilitating integration with other tools and APIs.

#### 4 Evaluation of REST-at

Our evaluation of REST-at was conducted in collaboration with Test Scouts. The company has fewer than 50 employees, most of whom are consultants contracted to various European companies. We answer our research questions in three stages. Figure 4 shows an overview of our methodology to evaluate REST-at. After deploying REST-at with different LLMs (Step 1), we perform two experiments to compare how different models and prompt templates affect the identification of trace links between requirements and tests.

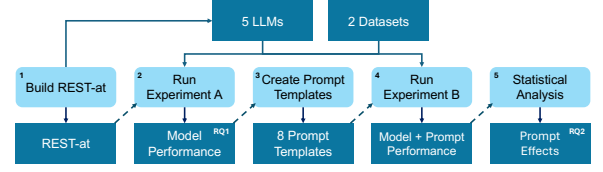


Figure 4: Overview of our methodology, including the activities conducted and corresponding deliverables.

Table 1: Example of a requirement and two associated test cases (TC).

| ID            | Content                                                                                                                                                                 |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Req 86</b> | The metering units shall be able to handle both primary and secondary registration of Metrics.                                                                          |
| <b>TC 430</b> | Verify that the Metering Units can handle primary registration of Measured Values. Verify that the Metering Units can handle secondary registration of Measured Values. |
| <b>TC 478</b> | Verify that the Metering Units can handle both primary and secondary recording of Metering Values.                                                                      |

#### 4.1 Dataset Collection

We used two datasets containing known trace links between requirements and tests, which acted as a ground truth for our experiments. The first dataset is an openly available specification from the Bluetooth Headset Profile 1.2 (BTHS);<sup>4</sup> while the second dataset was provided by Göteborg Energi (GBGE) an energy provider company that works with our industry partner. Table 1 shows an example of a requirement and its associated tests from the GBGE dataset.

Table 2 provides descriptive statistics of the datasets, including the number of artifacts and various types of trace links between requirements and tests. Our goal is to have a diverse configuration of trace links in our samples.

**Table 2: Relationships between requirements and tests, specifying RE:ST as one-to-one (1:1), one-to-many (1:M), many-to-one (M:1), and many-to-many (M:M) links. P is prevalence, NT is not tested. For instance, GBGE had 6 requirements that individually mapped to many tests cases (14 test artifacts linked to a single requirement).**

| DS   | $\sum RE$ | $\sum ST$ | 1:M  | M:M | M:1  | 1:1   | NT | P     |
|------|-----------|-----------|------|-----|------|-------|----|-------|
| GBGE | 40        | 41        | 6:14 | 0:0 | 10:5 | 22:22 | 2  | 2.8%  |
| BTHS | 8         | 16        | 3:8  | 3:6 | 0:0  | 2:2   | 1  | 16.7% |

The datasets differ from one another in size and prevalence. BTHS has fewer tests and requirements, as well as a higher *prevalence* than GBGE, i.e., the ratio of positives in the dataset. For instance,  $P = 16.7\%$  in BTHS means that out of all possible trace links, i.e.,  $8 \text{ requirements} \times 16 \text{ tests} = 120 \text{ trace links}$ , only 16.7% of them are actually present as links between requirements and tests.

<sup>4</sup><https://www.bluetooth.com/specifications/specs/headset-profile-1-2/>

It is important to note that a low prevalence of trace links is not necessarily a bad thing, especially for large datasets, because test cases are supposed to be specific and, therefore, should not cover many requirements.

## 4.2 Experimental Design

We designed two experiments (Experiment A and Experiment B) to examine the effects of two independent variables: the choice of LLMs and the design of prompt templates. Depending on the experiment, these variables serve either as **factors** or **control variables**. The first variable, LLMs, includes five levels corresponding to the models integrated with REST-at: GPT-3.5, GPT-4, LLAMA, MISTRAL, and MIXTRAL. The second variable, prompt templates, consists of eight alternatives, including a baseline from the literature [16]: BASE, PT1, PT2, PT3, PT4, PT5, PT6, and PT7. To keep the experimental design focused on decoder-only LLMs and reduce the number of treatments, we defer the inclusion of additional methods such as string distance techniques (e.g., Jaccard, Cosine similarity) and BERT-based models to future work.

We compare the trace links generated by the different LLMs against the ground truth to calculate our **dependent variables**: recall, precision, and  $F_1$ -score metrics. Each model was executed 30 times with each dataset to observe potential variations in the output data due to LLM's stochastic nature. We set the *temperature* parameter of the LLMs as 0.1 to reduce stochastic behavior in the models.<sup>5</sup>

**Experiment A - Comparing LLMs:** Experiment A evaluates RQ1, i.e., how a model impacts its ability to identify REST trace links, thus providing us with a baseline for Experiment B. Consequently, we control for the prompt variations by using a prompt template from the literature [16] (BASE). Our null hypothesis is that the different LLMs yield similar precision, recall and  $F_1$ .

**Experiment B - Analyzing Prompts:** We create several prompt templates using a variety of prompt engineering techniques to identify how those templates affect precision, recall, and  $F_1$  of the different LLMs. All prompts use Chain-of-Thought prompting, Chaining LLM Steps, and Few-Shot prompting, and follow a system and user prompt structure, where different elements/components from Figure 1 are used. Table 3 provides an overview of the prompt modifications, along with their ID. The complete set of prompt templates is available in our replication package [1].

## 4.3 Designing the Prompt Templates

Initially, the BASE prompt used a default role defined as “You are a helpful assistant.” and a structured JSON object as a default output format (DO). The first modification to the BASE prompt was to **only change the output format** to a list-based JSON array (PT1), aiming to reduce the GPT model's API usage cost and simplify the output format. To further explore the impact of meaningful modifications, we (i) varied the role specification (PT2 and PT3), (ii) reorganized the paragraphs of the prompt while utilizing all user prompt components (PT4 and PT5), and (iii) created a highly

structured prompt for the user prompt (“Task data” component) in conjunction with different output formats (PT6 and PT7).

**Table 3: Overview of the prompt modifications, each mapping to components in Figure 1. The modifications target: the Default Output format from BASE (DO), a New Output format (NO), Role Specific descriptions (RS), Reorganise Original Prompt (ReOP), and Prompt Patterns (PP).**

| ID          | Modification description                                                                                                                                                                     |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>BASE</b> | <b>DO:</b> System: Default Role. User: Instruction and Objectives, Context information, Rules and Guidelines, and Output format (JSON Object)                                                |
| <b>PT1</b>  | <b>NO:</b> System: The output format is changed to a JSON array                                                                                                                              |
| <b>PT2</b>  | <b>NO + RS:</b> Changed from default system prompt role to a task-specific role                                                                                                              |
| <b>PT3</b>  | <b>DO + RS</b>                                                                                                                                                                               |
| <b>PT4</b>  | <b>RS + NO + ReOP:</b> System: Tone guidelines as role, Instructions, Context information, Rules and Guidelines, and Output format. User: Task description, Task context and, Task data      |
| <b>PT5</b>  | <b>RS + DO + ReOP</b>                                                                                                                                                                        |
| <b>PT6</b>  | <b>RS + NO + PP:</b> Strict structure using section division of each component. Tone guidelines as role, Instruction and Objectives, Task context, Rules, and Output format. User: Task data |
| <b>PT7</b>  | <b>RS + DO + PP</b>                                                                                                                                                                          |

Below, we present an example illustrating the different components of the prompt PT4 (RS + NO + ReOP). The remaining prompts are available in our replication package.

### System Prompt:

Act as a mapping system, that receives a requirement and test cases and returns a list of the test cases that test that specific requirement.

It IS CRUCIAL that your response MUST adhere to the exact JSON format outlined below. Incorrect formatting may result in your response being improperly processed.

If any test cases are testing the requirement, answer ONLY with the test case ID(s) using this format for your report:

[ "<test ID 1>", "<test ID 2>", "<test ID 3>", ... ]

If no test cases are testing the requirement, use this format: []

### User Prompt:

Your task is to identify and return the test cases that directly validate the requirement

Here are the requirements: req  
and here are the test cases: tests

Please preserve the formatting and overall template specified above.

A full two-factorial design would result in 40 treatment combinations. To make the analysis more manageable, we apply minimum

<sup>5</sup>In LLMs, temperature values range from 0 to 2 and determine the randomness and creativity of the model's output. The higher the temperature, the more variation and creativity we can expect in the generated answer.

performance thresholds of 75% precision and 80% recall, excluding model-prompt combinations that produce unreliable outcomes. Since the interaction effects are not the focus of this study, we treat the LLM variable as a block. This results in a one-factor experimental design with multiple levels, such that we only compare the prompt variations *per LLM*, rather than across all LLMs. Our null hypothesis states that there is no significant difference in precision, recall, or  $F_1$  scores between the prompts for the same LLM.

#### 4.4 Data Analysis and Reproducibility

Experiments A and B follow a one-factor design with several levels. Due to the size of our sample and the fact that our results did not follow a normal distribution, we opted for non-parametric statistical tests. We begin with a Kruskal-Wallis test to check for differences between levels, and if the null hypothesis is rejected, we proceed with a post-hoc pairwise analysis of the different levels. We use the non-parametric Mann-Whitney test with a Bonferroni adjustment.<sup>6</sup>

Moreover, we measure the magnitude of the differences by using the non-parametric Vargha-Delaney (VDA)  $\hat{A}_{12}$  measure.  $\hat{A}_{12}$  quantifies the probability that, e.g., a randomly selected recall from a prompt will be greater than a randomly selected recall from another prompt. In our post-hoc analysis, we only discuss the treatments that showed statistically significant differences (SSD) and a large effect size from  $\hat{A}_{12}$ .

Our replication package [1] includes the source code of REST-at, as well as the evaluation scripts used to create the plots and tables, and the statistical analysis described here. The repository's README also includes the specific prompt template definitions, the exact model snapshots for the open-weight models, and the exact GPT models used.

## 5 Results

### 5.1 RQ1: Comparing LLMs for REST Alignment

The complete tables and results for Precision, Recall,  $F_1$ -score, and other measures (e.g., Accuracy) are available in our replication package [1]. For brevity, we focus on precision and recall to distinguish the impacts of false positives and false negatives, respectively. In our analysis, we search for a balance between high  $F_1$ , followed by high recall and, lastly, high precision. Our reasoning is that, from a tester's perspective, confirming whether an identified link is indeed correct (i.e., removing false positives) is less time-consuming than trying to find missed links (false negatives). Table 4 presents the results for each model and dataset.

For the BTHS dataset, Mistral and LLAMA 3 stand out with the highest average recall (85%), while achieving a high  $F_1$  score, indicating a robust balance between precision and recall. GPT-4 had the highest precision but a lower recall, indicating that it might miss relevant test cases. LLAMA 3 covered more test cases, with higher false positives, making it more effective overall. Therefore, despite GPT-4's higher precision, LLAMA 3's higher average recall makes it more effective. Additionally, GPT-4 and GPT-3.5 exhibit higher standard deviations, suggesting more variance in their performance. Lastly, Mistral had low results in all measures (below 65%).

<sup>6</sup>The Bonferroni adjustment is more conservative because it reduces the level of significance for each pairwise test by dividing the level of significance  $\alpha$  by the number of comparisons being made [8, 10].

**Table 4: Performance comparison of the five language models across two datasets with the average performance and standard deviation ( $\mu \pm \sigma$ ) for each model in terms of Precision, Recall, and  $F_1$ -score. Highlighted values represent the highest performance in each metric.**

| Dataset        | BTHS  |                   |                  |                  |                  |
|----------------|-------|-------------------|------------------|------------------|------------------|
|                | Model | GPT 3.5           | GPT 4            | LLAMA            | MISTRAL          |
| BTHS Precision |       | 62.4% $\pm$ 11.2% | 85.6% $\pm$ 1.5% | 73.9% $\pm$ 0.0% | 64.5% $\pm$ 1.2% |
| BTHS Recall    |       | 57.2% $\pm$ 15.4% | 60.0% $\pm$ 7.3% | 85.0% $\pm$ 0.0% | 55.0% $\pm$ 0.0% |
| BTHS $F_1$     |       | 57.3% $\pm$ 8.9%  | 70.4% $\pm$ 5.6% | 79.1% $\pm$ 0.0% | 59.4% $\pm$ 0.6% |
| Dataset        | GBGE  |                   |                  |                  |                  |
|                | Model | GPT 3.5           | GPT 4            | LLAMA            | MISTRAL          |
| GBGE Precision |       | 97.9% $\pm$ 0.7%  | 98.5% $\pm$ 1.2% | 92.0% $\pm$ 0.0% | 94.8% $\pm$ 0.0% |
| GBGE Recall    |       | 92.6% $\pm$ 1.7%  | 95.7% $\pm$ 0.4% | 100% $\pm$ 0.0%  | 91.3% $\pm$ 0.0% |
| GBGE $F_1$     |       | 95.2% $\pm$ 0.8%  | 97.1% $\pm$ 0.6% | 95.8% $\pm$ 0.0% | 93.0% $\pm$ 0.5% |

For the GBGE dataset, Mistral and LLAMA 3 showed consistent performance. LLAMA 3 achieved a 100% recall score, ensuring comprehensive requirement testing. Mistral stood out with the highest  $F_1$ -score (97.8%), demonstrating reliable requirement testing. Nonetheless, results for all models were above 90% which means that, in practice, *any* model would be a suitable candidate for REST alignment in their dataset. The GBGE dataset originates from our industry partner, where standardized guidelines for documenting requirements and tests help ensure a higher baseline quality of the text descriptions.

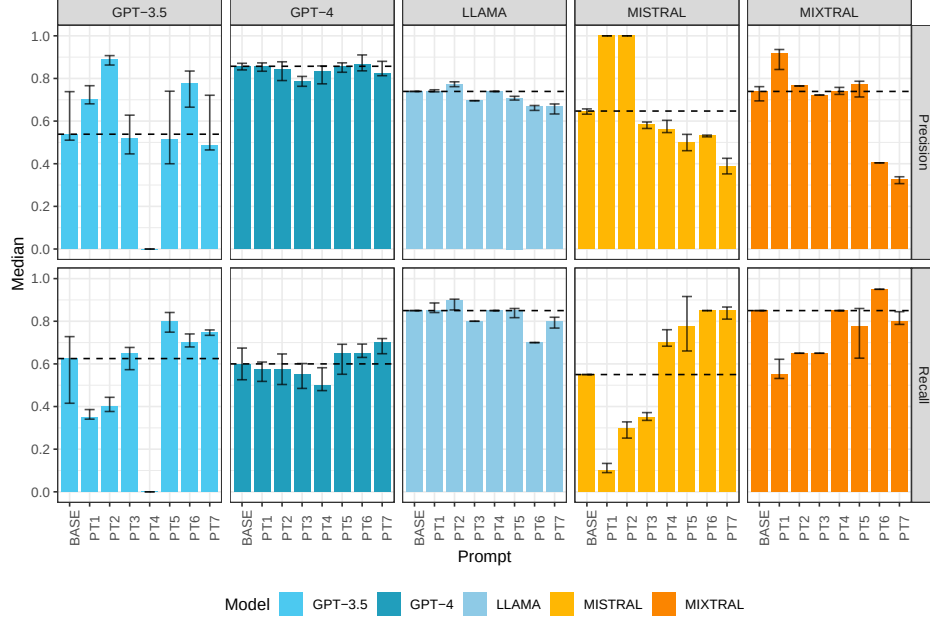
**RQ1:** Our results suggest that model size has no clear practical impact on effectiveness in REST alignment, as precision, recall, and  $F_1$  scores were generally high across models, with the exception of GPT-3.5 and Mistral (BTHS dataset). The medium-size models of this study, Mistral and LLAMA 3, were the most reliable for comprehensive REST alignment.

### 5.2 RQ2: Comparing Prompt Templates

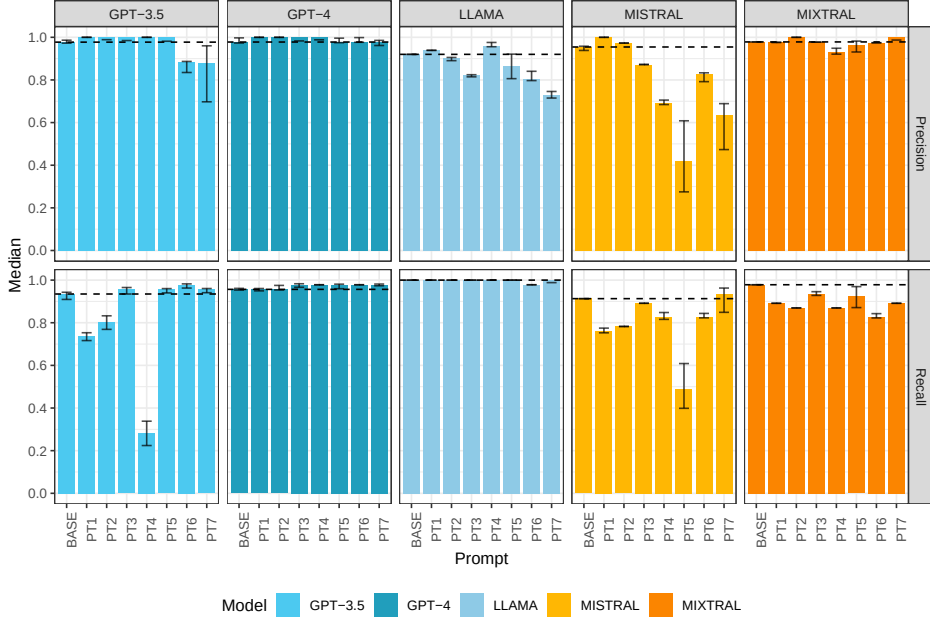
For RQ2, we investigate how varied prompt structures affect each model. As seen in Figure 5, each model responded distinctly to the different prompts. LLAMA 3 showed consistent recall for the GBGE dataset, aligning with GPT-4's best average recall while maintaining relatively high precision across different prompts. LLAMA, together with PT2, demonstrated the highest recall for BTHS data across all models, but with a decrease in precision in the GBGE dataset compared with the results from the BASE prompt.

GPT models are more sensitive to the specific structure of the prompts. For instance, in the GBGE dataset, GPT-3.5 with PT5 shows the highest standard deviation — 12.91% in precision. In contrast, open-weight models show much lower stochastic behavior.

Figure 5 also shows complex performance variations between models and prompts, indicating trade-offs. A Kruskal-Wallis test revealed statistically significant differences (SSD) between prompts for all models and datasets ( $p < 0.001$ ). Therefore, we conducted post-hoc pairwise tests to identify specific differences. As outlined in Section 4, we only compared prompt templates with  $P \geq 0.75$  and  $R \geq 0.8$ . Only one prompt from the BTHS dataset met this threshold (LLAMA using PT2), compared to 32 combinations of prompts and models for GBGE. This reduced our statistical analysis from 140 pairwise tests to 94.



(a) BTHS



(b) GBGE

**Figure 5: Median precision and recall with standard deviation for the 5 models using the 8 prompts on 5a) BTHS and 5b) GBGE. The dashed lines emphasise how the different prompts affected the BASE prompt.**

For brevity, Table 5 reports the number of times each prompt showed a statistically significant difference and a large effect during the pairwise comparisons. Below, we summarize the prompts that showed statistically significant improvements in precision and recall for each model in the GBGE Dataset.

- GPT-3.5 showed higher variance and a tied improvement between PT3 and PT5. We choose PT5 (DO + RS + ReOP) as it showed improved precision and recall in BTHS.
- GPT-4 showed improved precision and recall with PT4 (NO + RS + ReOP).

**Table 5: Summary of our pairwise posthoc analysis on the GBGE runs. The table shows how many times each prompt showed better precision/recall with a statistically significant difference and a large effect size on Precision (P) and Recall (R). The highlighted cells show the best performing model-prompt pair. In case of a tie, we choose the prompt that also showed improvements in the BTHS dataset. Model-prompt pairs that did not reach the precision and recall threshold for Experiment B are excluded.**

| Prompt       | GPT-3.5 |    | GPT-4 |    | LLAMA |   | MISTRAL |   | MIXTRAL |    |
|--------------|---------|----|-------|----|-------|---|---------|---|---------|----|
|              | P       | R  | P     | R  | P     | R | P       | R | P       | R  |
| BASE         | 2       | 1  | –     | –  | 4     | 1 | 2       | 2 | 5       | 7  |
| PT1          | –       | –  | 4     | –  | 5     | 1 | –       | – | 2       | 3  |
| PT2          | 3       | –  | 4     | –  | 2     | 1 | –       | – | 6       | 1  |
| PT3          | 3       | 2  | 2     | 3  | –     | 1 | 1       | 1 | 3       | 5  |
| PT4          | –       | –  | 4     | 3  | 6     | 1 | –       | – | –       | 1  |
| PT5          | 3       | 2  | –     | 2  | 1     | 1 | –       | – | 1       | 3  |
| PT6          | –       | 5  | 1     | 3  | –     | – | –       | – | 1       | –  |
| PT7          | –       | 2  | –     | 3  | –     | – | –       | – | 6       | 3  |
| <b>Total</b> | 11      | 12 | 15    | 14 | 18    | 6 | 3       | 3 | 24      | 23 |

- LLAMA demonstrated consistent performance improvements with PT4 (NO + RS + ReOP).
- MISTRAL and MIXTRAL: Both models showed the best results with the BASE prompt.

These observations underscore the varied impacts of different prompt modifications on model performance across various datasets and models, highlighting the sensitivity of those models to small changes in prompts. MIXTRAL and MISTRAL were not significantly affected by the prompt changes. PT4 and PT5 have shown improvements for GPT-3.5, GPT-4, and LLAMA. These prompt templates share the inclusion of a role specification (RS) and a structured organization, highlighting the benefits of these elements. These observations are in line with recent research indicating that the use of personas or few-shot examples in prompts can lead to variations in model performance during code generation [18].

When comparing the results of BASE and PT1, the models exhibited a trade-off where declines in average recall for GPT-3.5 and Mistral were often accompanied by improvements in precision. For example, as shown in Figure 5, the average precision of GPT-3.5 in BTHS increased. Mistral, on the other hand, maintained an average precision of 100% but saw a decrease in average recall from 55% to 11.17%. LLAMA demonstrates robustness across both output formats and datasets.

Although PT4 and PT5 differ in how they present the output, we noticed a surprising effect on the output change for other models, particularly for GPT-3.5. To investigate the outlier behavior of GPT-3.5 with PT4 on the BTHS dataset, we conducted a series of controlled experiments with various prompt configurations and temperature settings, as summarized in Table 6. These configurations were designed to isolate and identify the factors contributing to the zero performance metrics. Changing the temperature is a useful way to investigate outlier behavior because it influences the randomness of the model’s outputs. Lower temperatures produce more deterministic outputs, while higher temperatures introduce more variability. This can help reveal whether the zero performance

metrics are due to the model’s deterministic behavior or other factors.

**Table 6: Complementary runs on variations of PT4 using GPT-3.5**

| Prompt         | Temperature | Recall     | Precision   | $F_1$ -score |
|----------------|-------------|------------|-------------|--------------|
| NO + ReOP + RS | 0.1         | 0.0 ± 0.0  | 0.0 ± 0.0   | 0.0 ± 0.0    |
| NO + ReOP + RS | 0.2         | 0.0 ± 0.0  | 0.0 ± 0.0   | 0.0 ± 0.0    |
| NO + ReOP + RS | 0.7         | 0.0 ± 0.0  | 0.0 ± 0.0   | 0.0 ± 0.0    |
| NO + ReOP + RS | 1           | 0.0 ± 0.0  | 0.0 ± 0.0   | 0.0 ± 0.0    |
| NO + ReOP      | 0.1         | 3.0 ± 6.0  | 20.0 ± 40.0 | 5.2 ± 10.4   |
| NO + ReOP      | 0.2         | 6.0 ± 3.7  | 70.0 ± 40.0 | 11.0 ± 6.8   |
| NO + ReOP      | 1           | 30.0 ± 0.0 | 100.0 ± 0.0 | 46.2 ± 0.0   |

NO + RS + ReOP, temperature 0.1 was queried 10 times, while the rest only had 5 inquiries.

The experiments revealed that GPT-3.5’s outlier behavior with PT4 is related to the specific prompt combination of instructions rather than the model’s temperature setting, since adjusting the prompt configuration severely impacted the performance metrics. This is also supported by the performance of GPT-3.5 in combination with PT4 in the GBGE dataset, where average recall drops by 28.12% compared with its BASE performance.

Additionally, the change in output format played a significant role in improving GPT-3.5’s performance in the BTHS dataset. By comparing PT4 and PT5 in Figure 5, GPT-3.5 went from 0 precision and recall 57.0% and 79.5%, respectively. The only difference between those prompts was changing the JSON list output format to JSON object (PT5-DO + RS + ReOP). Due to the lack of transparency regarding the model itself, we could not identify a root cause for this outlier behaviour.

**RQ2:** Different prompt structures impact an LLM’s ability to identify trace links, with each model responding uniquely, indicating a high sensitivity to prompt changes. Although precision varies, LLAMA consistently achieves 100% average recall with most prompts on the GBGE dataset.

**Output Format:** Simplifying the output format reveals an unexpected sensitivity not only to the input prompt but also to the required output format. This highlights a challenge in transferability. LLAMA exhibits robustness by maintaining or improving its average recall.

## 6 Discussion

Here, we discuss the implications of our findings. Particularly, we relate our conclusions to existing literature on the topic, raise some technical challenges and recommendations for researchers aiming to work with LLMs. We also suggest focus areas for advancing research and practice in LLMs applied to REST alignment.

### 6.1 Implications of Prompt Engineering

While we agree with Zheng et al. [38] on the challenge of identifying optimal roles, *our RQ2 results do not support their argument that role specification consistently improves performance*. Instead, our eight

prompt variations showed different trade-offs for all LLMs due to the change of output format, the introduction of task-specific roles, and the restriction of prompts.

Our results from RQ2 resonate with Marvin et al.'s argument on the critical role of the model's architecture and training data when choosing an LLM [22]. Note that the sensitivity seen in RQ2 applies to (i) different versions of GPT models, and (ii) models with similar training data but different architectures and sizes (e.g., Mistral and Mixtral). In short, each model responds uniquely to different prompt structures, underlining the necessity of crafting prompts tailored to specific models.

These findings also indicate a *transferability challenge* and align with arguments from Bush et al.'s [6] on the importance of understanding to what extent a selected prompt is specific to an LLM. Given the high sensitivity of LLMs to small variations in input, we argue that future research should prioritize the development of systematic strategies for prompt evaluation. Khojah et al. made similar efforts by applying structured combinations of prompt engineering techniques to code generation. Their results showed that combining multiple techniques does not necessarily lead to better performance compared to using some techniques individually (e.g., personas or few-shot) [18].

**Recommendation:** The investigated models were highly sensitive to prompt variations with both positive and negative impacts, depending on the model and prompt. Consequently, the research community should focus on identifying strategies to evaluate prompts systematically.

## 6.2 Quality of Requirements and Test Artefacts

Common errors in the results often arise from ambiguous requirements and test case specifications. One particular example is the trace link between Req-4.2.1 to Test-HSP/HS/IAC/BV-01-I, we consulted with four domain experts from our industry partner who confirmed its ambiguity. Note that, in our experiments, we did not provide any IDs mapping requirements to the models, i.e., we removed them to assess the extent of using the artifact's content in creating trace links.

Ambiguity in requirements hinders both developers and LLMs, which results in REST misalignment and thereby can decrease software quality and reliability [3, 19, 36]. Clarifying requirements improves the contextual mapping between test cases and requirements, enhancing automated trace link mapping. Besides improving traceability, standardized documentation ensures consistent understanding among team members, improving contextual integrity.

**Recommendation:** Improve REST artifacts by clarifying requirements, test descriptions, and simplifying IDs. This minimizes the need to clean or pre-process the data, thus enhancing automated trace link mapping and making AI tool integration in development more scalable and effective.

## 6.3 Development Tools and Constraints

To solve the hardware requirements and cost challenges, we explored GGUF (GPT-Generated Unified Format), a *quantization* method designed to store and run LLMs on CPUs effectively. Quantization reduces the precision of the model's weights, which saves memory and speeds up the inference, the ability of the model to generate responses at the cost of quality loss [15]. Applications such as LM Studio and Ollama<sup>7</sup> are good alternatives to run quantized models locally with a lower technical knowledge need, thereby a suitable environment for initial practical exploration. Due to the trade-off, we opted to utilize cloud resources over GGUF, which allows a fair comparison of the models. Nonetheless, future iterations of REST-at will incorporate quantized models to examine whether the reduction in computational cost compensates for any loss in performance.

Regarding the development of REST-at, our adoption of Streamlit for fast prototyping significantly enhanced our development process. Streamlit allowed us to quickly integrate our proof-of-concept Python scripts into a web application which helped us allocate more time to tackling the steep learning curve of configuring the LLMs. This experience emphasizes how a careful selection of development tools can accelerate software development and encourage an adaptable learning environment.

**Recommendation:** There is a need for developing scalable deployment frameworks for open-weight models that optimize GPU usage, and minimize operational costs. Choose development tools that enable rapid prototyping (e.g., Streamlit), as this can accelerate implementation and free up time to address more complex technical challenges.

## 6.4 Lack of Transparency and Standardization

Lack of transparency about training data, and development processes remain a challenge in open-weight model documentation and academic literature. Essential details about datasets and parameter configurations (e.g., temperature) are often omitted, decreasing the reproducibility and reliability of results [28].

For instance, Meta's LLAMA 3 hardware requirements are not disclosed in any documentation. The importance of prompt structure for Mixtral and Mistral was only mentioned in a small note on MistralAI's webpage<sup>8</sup>. We discovered important configuration details, such as the mask attention parameter for Mistral and Mixtral, through warning messages during our experiment. Additionally, using `model.eval()` (equivalent to `model.train(False)`) is important to prevent PyTorch modules (e.g., Dropout and BatchNorm2d) from behaving differently during evaluation, a responsibility that falls on the user<sup>9</sup>.

The lack of standardization, accessibility, and centralized knowledge in prompt engineering, combined with a lack of transparency about dataset specifications, forces developers to rely on trial-and-error methods for optimal configurations. This hinders development and prevents the full utilization of the model's capabilities.

<sup>7</sup><https://lmstudio.ai/> and <https://ollama.com/>

<sup>8</sup>Mistral: [https://docs.mistral.ai/getting-started/open\\_weight\\_models](https://docs.mistral.ai/getting-started/open_weight_models)

<sup>9</sup><https://pytorch.org/docs/stable/notes/autograd.html>

**Recommendation:** Prioritize using `model.eval()` even if you do not know which specific modules are used in the model. This will ensure that all such modules switch to evaluation mode, preventing unintended behavior and ensuring consistent results.

Different terms (e.g., system and user prompts) are used interchangeably by researchers and practitioners without clear definitions. Consistent with the challenges identified by Bommasani et al. [4], our findings highlight the need for LLM developers and companies to improve their documentation by clarifying ambiguous language, centralizing existing information sources, and providing more details about the required configuration and their consequences.

Our experience concurs with Miller [23] and Liesenfeld et al. [21] regarding the importance of openness and thorough documentation on LLMs. Openness enables a better understanding of how the models work, facilitating enhancements and adaptations while increasing insight into the model's strengths and limitations. This not only fosters innovation and allows for precise modifications to improve performance or adapt to new uses, but also increases reproducibility and avoids redundancy in research.

Furthermore, the need for transparency also extends to experiments with LLM tools. For example, while Sawada et al. [29] made significant contributions with their encoder-based REST alignment tool, the paper did not disclose details about the dataset used for training or evaluation. Additionally, their evaluation focuses on accuracy (of 75%) without providing other metrics (e.g., recall, or true positives). Therefore, we cannot compare REST-at with their proposed tool to identify performance differences between encoder-based and decoder-only applications for REST alignment.

**Recommendation:** When integrating LLMs into evaluations or tools, prioritize improving documentation, centralizing information sources, and detailing configuration requirements.

## 6.5 Threats to Validity

Based on Sallou et al. [28] guidelines, we identify the following threats to validity from involving LLMs in our study. The *unpredictability in model evolution* can lead to inconsistencies in results, while the non-deterministic nature of LLMs poses challenges for reproducibility. We mitigate this threat by performing multiple runs and sharing execution metadata for each experiment, such as the system fingerprint (version), seed, date of query, prompts, temperature, and other configuration details. That information, along with our analysis scripts, is available in our data package.

Another concern is selection bias; the non-proprietary LLMs chosen (Mistral 7B, Mixtral 8x7B, and LLAMA 3 70B) may not represent all non-proprietary models available. In this case, we have selected the LLMs that have shown comparative performance with OpenAI's LLMs across multiple benchmarks. The non-deterministic nature of LLMs introduces an internal validity threat. We mitigate this threat by comparing the outputs generated by the new prompt and the previous prompt used, aiming to identify patterns and

correlations. From a setup perspective, we use a temperature value of 0.1, which reduces the randomness of the model's output.

Due to computing limitations, the samples from the execution of different prompts vary in size and are relatively small. Despite these limitations, we observed a relatively low or negligible standard deviation. Moreover, we follow empirical guidelines related to statistical analysis, where we perform distribution tests and corrections to mitigate threats to conclusion validity [10]

We foster construct validity in our choice of models. Although limited computing resources from C3SE constrained the number of models we could evaluate, we include five models from three different companies to complement the existing literature, which predominantly focuses on GPT models from OpenAI.

Our research is based on the tester's perspective due to data availability and industry partners' expertise. While RE-related threats are not our focus, the requirements artifacts analyzed are from existing, deployed software systems. Lastly, our findings stem from two datasets, which limits the generalization of our studies to different software projects or domains. We aim to replicate the experiments with more models, prompt templates and datasets in future studies.

## 7 Conclusion

We introduce REST-at, a tool to create traceability between requirements and system-level tests using different proprietary and open-weight LLMs. The open-weight models allow practitioners to host REST-at locally for better data governance. REST-at reduces the time-consuming task of manual traceability and effectively addresses collaborative challenges in REST alignment. Moreover, we evaluate REST-at in two experiments that compare how different models and prompt structures affect REST-at in identifying correct trace links. All LLMs produced acceptable trace links, but our statistical analysis reveals differences in performance regarding the combinations of models and prompts. In short, our key findings are:

- **Model performance:** The medium models of this study, like Mixtral 8x7B (56B), and larger models like LLAMA 3 70B, performed well in aligning test cases with requirements, challenging the assumption that larger models are always superior.
- **Challenges with Open-Weight models:** We highlight significant challenges with open-weight models, including high hardware demands, data governance challenges, transparency, and the need for standardized documentation.
- **Complexity of Prompt Engineering:** The high sensitivity of LLMs to prompts highlights the often underestimated complexity of prompt engineering. It involves a complex and time-consuming process requiring technical expertise and model-specific knowledge.

Future work should prioritize systematic prompt evaluation, developing prompt techniques for open-weight models, and enhancing documentation to improve LLMs' reliability in software traceability. Additionally, REST-at can be improved by adding features that enhance communication between testers and requirements engineers, reducing misinterpretations and leading to higher quality software that meets users' expectations [36]. Transparent

documentation and improved communication will foster better understanding, innovation, and reproducibility.

## Acknowledgments

We thank our industry partner, Test Scouts, for their insights, data, and continued mentoring throughout this research. LLM executions were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement no. 2022-06725. This work was also supported by the Lars Pareto Grant from the University of Gothenburg.

## References

- [1] Nicole Leon-Quinsted, Bao Lindgren, Mert Yurdakul, Francisco Gomes de Oliveira Neto. 2024. REST-at replication package. <https://github.com/BaoLindgren/REST-at/>.
- [2] Anthropic. [n. d.]. System prompts. <https://docs.anthropic.com/claude/docs/system-prompts>. [Accessed: 2024-03-29].
- [3] Elizabeth Bjarnason, Per Runeson, Markus Borg, Michael Unterkalmsteiner, Emelie Engström, Björn Regnell, Giedre Sabaliauskaite, Annabella Loconsole, Tony Gorschek, and Robert Feldt. 2014. Challenges and practices in aligning requirements with verification and validation: a case study of six companies. *Empirical software engineering* 19 (2014), 1809–1855.
- [4] Rishi Bommasani, Kevin Klyman, Shayne Longpre, Sayash Kapoor, Nestor Maslej, Betty Xiong, Daniel Zhang, and Percy Liang. 2023. The foundation model transparency index. *arXiv preprint arXiv:2310.12941* (2023).
- [5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.
- [6] Kiran Busch, Alexander Rochlitz, Diana Sola, and Henrik Leopold. 2023. Just Tell Me: Prompt Engineering in Business Process Management. In *Enterprise, Business-Process and Information Systems Modeling*, Han van der Aa, Dominik Bork, Henderik A. Proper, and Rainer Schmidt (Eds.). Springer Nature Switzerland, Cham, 3–11.
- [7] C3.ai. 2023. Explainability - Machine Learning Glossary. <https://c3.ai/glossary/machine-learning/explainability/>. [Accessed: 2024-02-09].
- [8] Shi-Yi Chen, Zhe Feng, and Xiaolian Yi. 2017. A general introduction to adjustment for multiple comparisons. *Journal of thoracic disease* 9, 6 (2017), 1725.
- [9] Aron Culotta and Nicholas Mattei. 2023. Use Open Source for Safer Generative AI Experiments. *MIT Sloan Management Review* 65, 1 (2023), 1–3.
- [10] Francisco Gomes de Oliveira Neto, Richard Torkar, Robert Feldt, Lucas Gren, Carlo A Furia, and Ziwei Huang. 2019. Evolution of statistical analysis in empirical software engineering research: Current state and steps forward. *Journal of Systems and Software* 156 (2019), 246–267.
- [11] Katherine R Dearnstyn. 2025. Intelligent Traceability to Support Software Maintainability and Accountability. In *2025 IEEE 33rd International Requirements Engineering Conference (RE)*. IEEE, 607–611.
- [12] Christof Ebert and Panos Louridas. 2023. Generative AI for Software Practitioners. *IEEE Software* 40, 4 (2023), 30–38. <https://doi.org/10.1109/MS.2023.3265877>
- [13] Sabit Ekin. 2023. Prompt engineering for ChatGPT: a quick guide to techniques, tips, and best practices. *Authorea Preprints* (2023).
- [14] Michael Felderer, Eduard Paul Enoiu, and Sahar Tahvili. 2023. Artificial Intelligence Techniques in System Testing. In *Optimising the Software Development Process with Artificial Intelligence*. Springer, 221–240.
- [15] Hugging Face, Inc. 2024. Hugging Face Hub. [https://huggingface.co/docs/optimum/concept\\_guides/quantization](https://huggingface.co/docs/optimum/concept_guides/quantization). [Accessed: 2024-03-07].
- [16] Samuel Ivarsson and Jesper Setterström. 2023. Automated Alignment of Software Requirements and Test Cases.
- [17] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Léo Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. *arXiv:2310.06825 [cs.CL]* [Accessed: 2024-03-07].
- [18] Ranim Khojah, Francisco Gomes de Oliveira Neto, Mazen Mohamad, and Philipp Leitner. 2025. The Impact of Prompt Programming on Function-Level Code Generation. *IEEE Transactions on Software Engineering* 51, 8 (2025), 2381–2395. <https://doi.org/10.1109/TSE.2025.3587794>
- [19] Johanna Kukkanen, Kyösti Väkeväinen, Marjo Kauppinen, and Eero Uusitalo. 2009. Applying a Systematic Approach to Link Requirements and Testing: A Case Study. In *2009 16th Asia-Pacific Software Engineering Conference*. 482–488. <https://doi.org/10.1109/APSEC.2009.62>
- [20] Learn Prompting. [n. d.]. Learn Prompting: Your Guide to Communicating with AI — [learnprompting.org](https://learnprompting.org/docs/basics/formalizing). <https://learnprompting.org/docs/basics/formalizing>. [Accessed: 2024-04-01].
- [21] Andreas Liesenfeld, Alianda Lopez, and Mark Dingemanse. 2023. Opening up ChatGPT: Tracking openness, transparency, and accountability in instruction-tuned text generators. In *Proceedings of the 5th International Conference on Conversational User Interfaces* (<conf-loc>, <city>Eindhoven</city>, <country>Netherlands</country>, </conf-loc>) (CUI '23). Association for Computing Machinery, New York, NY, USA, Article 47, 6 pages. <https://doi.org/10.1145/3571884.3604316>
- [22] Ggaliwango Marvin, Nakayiza Hellen, Daudi Jjingo, and Joyce Nakatumba-Nabende. 2024. Prompt Engineering in Large Language Models. In *Data Intelligence and Cognitive Informatics*, I. Jeena Jacob, Selwyn Piramuthu, and Przemyslaw Falkowski-Gilski (Eds.). Springer Nature Singapore, Singapore, 387–402.
- [23] Kyle Miller. 2024. Open Foundation Models: Implications of Contemporary Artificial Intelligence. <https://cset.georgetown.edu/article/open-foundation-models-implications-of-contemporary-artificial-intelligence/>. [Accessed:2024-05-10].
- [24] OpenAI. [n. d.]. Prompt engineering - OpenAI API. <https://platform.openai.com/docs/introduction>. [Accessed: 2024-09-03].
- [25] Yunjian Qiu and Yan Jin. 2024. ChatGPT and finetuned BERT: A comparative study for developing intelligent design support systems. *Intelligent Systems with Applications* 21 (2024), 200308. <https://doi.org/10.1016/j.iswa.2023.200308>
- [26] Sunil Ramlochan. [n. d.]. System Prompts in Large Language Models — [promptengineering.org](https://promptengineering.org/system-prompts-in-large-language-models/). <https://promptengineering.org/system-prompts-in-large-language-models/>. [Accessed 04-06-2024].
- [27] Tyler Richards. 2021. *Getting Started with Streamlit for Data Science: Create and deploy Streamlit web applications from scratch in Python*. Packt Publishing Ltd.
- [28] June Sallou, Thomas Durieux, and Annibale Panichella. 2024. Breaking the Silence: the Threats of Using LLMs in Software Engineering. In *Proceedings of the 2024 ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results* (Lisbon, Portugal) (ICSE-NIER'24). Association for Computing Machinery, New York, NY, USA, 102–106. <https://doi.org/10.1145/3639476.3639764>
- [29] Kae Sawada, Marc Pomerantz, Gus Razo, and Michael W. Clark. 2023. Intelligent requirement-to-test-case traceability system via Natural Language Processing and Machine Learning. In *2023 IEEE 9th International Conference on Space Mission Challenges for Information Technology (SMC-IT)*. 78–83. <https://doi.org/10.1109/SMC-IT56444.2023.00017>
- [30] Saurabh Shukla, Arushi Maheshwari, and Prashant Johri. 2021. Comparative Analysis of ML Algorithms and Stream Lit Web Application. In *2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*. 175–180. <https://doi.org/10.1109/ICAC3N53548.2021.9725496>
- [31] Sahar Tahvili, Leo Hatvani, Michael Felderer, Wasif Afzal, and Markus Bohlin. 2019. Automated Functional Dependency Detection Between Test Cases Using Doc2Vec and Clustering. In *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)*. 19–26. <https://doi.org/10.1109/AITest.2019.00-13>
- [32] Hamza Tamaouil, Mahmoud El Hamloui, and Mahmoud Nassar. 2024. Prompt Engineering: User Prompt Meta Model for GPT Based Models. In *Artificial Intelligence, Data Science and Applications*, Yousef Farhaoui, Amir Hussain, Tanzila Saba, Hamed Taherdoost, and Anshul Verma (Eds.). Springer Nature Switzerland, Cham, 428–433.
- [33] Michael Unterkalmsteiner, Robert Feldt, and Tony Gorschek. 2014. A taxonomy for requirements engineering and software test alignment. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 23, 2 (2014), 1–38.
- [34] Michael Unterkalmsteiner, Tony Gorschek, Robert Feldt, and Eriks Klotins. 2015. Assessing requirements engineering and software test alignment—Five case studies. *Journal of Systems and Software* 109 (2015), 62–77. <https://doi.org/10.1016/j.jss.2015.07.018>
- [35] U.S. Department of Commerce. [n. d.]. NTIA Solicits Comments on Open-Weight AI Models — [commerce.gov](https://www.commerce.gov/news/press-releases/2024/02/ntia-solicits-comments-open-weight-ai-models). <https://www.commerce.gov/news/press-releases/2024/02/ntia-solicits-comments-open-weight-ai-models>. [Accessed:2024-02-27].
- [36] Eero J. Uusitalo, Marko Komssi, Marjo Kauppinen, and Alan M. Davis. 2008. Linking Requirements and Testing in Practice. In *2008 16th IEEE International Requirements Engineering Conference*. 265–270. <https://doi.org/10.1109/RE.2008.30>
- [37] Jules White, Sam Hays, Quichen Fu, Jesse Spencer-Smith, and Douglas C Schmidt. 2023. Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design. *arXiv preprint arXiv:2303.07839* (2023).
- [38] Mingqian Zheng, Jiaxin Pei, and David Jurgens. 2023. Is "A Helpful Assistant" the Best Role for Large Language Models? A Systematic Evaluation of Social Roles in System Prompts. *arXiv:2311.10054 [cs.CL]* [Online]. Available: <https://doi.org/10.48550/arXiv.2311.10054>