Test Case Review Agent

Michael Ruggiero
michael.1.ruggiero@lmco.com

February 20, 2025

Contents

1 Executive Summary

Engineers developing test cases for applications and websites rely on Excel sheets to document test steps. However, these sheets are often poorly organized, inconsistently formatted, and difficult to search through. Engineers waste significant time locating relevant files, compiling fragmented steps, and clarifying missing details with colleagues. The absence of a structured database further exacerbates these inefficiencies.

To address these challenges, we propose an automated agent that streamlines test case discovery and refinement. The system follows a structured workflow based on five core processes, known as the **5 C's**:

Process	**Description**
Clean	Preprocess Excel sheets by removing inconsistencies, missing values, and irrelevant data. Only structured and usable test steps move forward.
Compile	Aggregate relevant test steps from different Excel sheets into a single structured list. Extract key columns and organize them efficiently.
Compare	Identify duplicate, conflicting, or incomplete test steps. Detect vague instructions and flag missing details for review. Leverages LLMs like Mixtral and Llama to assess clarity.
Coordinate	Connect engineers with relevant contacts for clarifications by extracting metadata such as authorship and last-modified details.
Clean	Ensure final test cases are refined, structured, and free of redundant or unclear steps. While a relational database would be ideal for maintaining test cases, the current approach compiles XLSX files into a dictionary-based format for efficient parsing and retrieval. Future iterations of this system could include database integration.

2 Problem Statement

TOSCA engineers rely on Excel sheets to construct test cases for applications and websites. However, these sheets are often **poorly stored, inconsistently formatted, and difficult to search through**. Engineers spend excessive time locating the relevant sheets and extracting useful information. Since SMEs do not completely understand the testing process they do not know the challenges in writing clean and actionable test.

Therefore, even when the appropriate sheets are found, the **test steps provided are frequently vague, incomplete, or not actionable**. Engineers are often required to manually reach out to the original authors or last editors to clarify missing details, introducing further delays in the test creation process.

Compounding these inefficiencies, **there is no structured database** to store and index these test cases. Instead, engineers must manually search through an ever-growing collection of spreadsheets, resulting in wasted time and inconsistent test quality.

To address these challenges, we propose an **automated agent** that will preform the **5 Cs**; **clean, compile, compare, coordinate and clean** test case data. This system will work along side engineers to retrieve relevant Excel sheets based on their test case description, extract and refine actionable steps, and identify key contacts for further clarifications—thereby significantly reducing the time required to generate high-quality test cases.

Figure 1: The 5 Ds of Dodgeball

3 Clean

Before any meaningful test case compilation can occur, the raw data from Excel sheets must be processed to remove inconsistencies, missing values, and redundant information. The current dataset consists of **1109 Excel sheets**, many of which contain **empty fields, non-standard formats, and irrelevant information**. Engineers often waste time manually filtering out unusable data, leading to inefficiencies.

The proposed workflow will automate this process by systematically scanning and cleaning the Excel sheets. The primary focus will be on identifying sheets labeled **"Test Case"**, as these contain structured columns: **Step Number**, **Step Description**, and **Step Expected Result**. Any sheet that does not conform to this structure will be ignored, significantly reducing the noise in the dataset.

Figure 2: Distribution of Excel sheets by type

Since there is no existing relational database, the initial cleaning and processing will be handled programmatically using **Python dictionaries** and **Pandas DataFrames**. While a structured database would be preferable, these data are not unmanageable, and the system will compile all **XLSX files into a dictionary-based format**, allowing for parsing and retrieval. This approach enables **lightweight indexing and transformation of data** without requiring an extensive database setup.

It is important to note that **the agent itself will not perform this initial cleaning**. Instead, the preprocessing pipeline will ensure that the data is structured before the agent engages with it. The agent will only process and refine the cleaned data, focusing on test case compilation, comparison, and coordination.

By automating the cleaning process outside the agent's core functionality, the workflow ensures that only high-quality, hallucination free data moves forward, reducing manual effort and improving efficiency in test case development.

Figure 3: Frequency of valid vs. invalid sheets

4 Compile

Once the raw data has been cleaned, the next step is to compile relevant test case information from multiple Excel sheets into a single structured list. Engineers currently spend excessive time manually identifying and extracting relevant test steps across disorganized files. The proposed system automates this process, ensuring that all necessary steps are consolidated efficiently.

The agent will scan all available Excel sheets and extract structured data from those labeled **"Test Case"**. These sheets typically contain three key columns: **Step Number**, **Step Description**, and **Step Expected Result**. By focusing exclusively on these structured fields, the system eliminates extraneous data and ensures that only useful information is included.

Figure 4: Frequency of valid vs. invalid sheets

Since no relational database is available, the system will compile all test case data into a **searchabge dictionary-based format using Python and Pandas**. Each test case will be indexed by its relevant metadata, allowing for efficient retrieval and organization. While this method allows for flexible storage and quick access, future improvements may include database integration for long-term scalability.

After extraction based upon a keyword prompt, the system aggregates all relevant test steps into a unified list, ensuring that engineers no longer need to manually compile data from multiple sources. Additionally, duplicate steps will be identified, and conflicting instructions will be flagged for review, improving overall test consistency.

The compiled test cases will be structured in a standardized format, making it easier for engineers to review, modify, and execute them. This ensures a more streamlined workflow, reducing manual effort and minimizing errors in test case development.

5 Compare

After compiling test steps from multiple sources, the next critical process is to compare them for consistency, completeness, and clarity. Engineers frequently encounter duplicate steps, conflicting instructions, and vague descriptions, which can lead to ambiguity and inefficiencies during test execution. The system automates this review process to improve the overall quality of test cases.

The system performs comparison at three levels:

- **Duplicate Detection:** Identifies redundant test steps that appear across multiple Excel sheets. If identical or near-identical steps exist, the system consolidates them to prevent unnecessary repetition.

- **Conflict Resolution:** Flags test steps that contradict one another. For instance, if two steps describe different expected results for the same action, they will be highlighted for engineer review.

- **Specificity & Actionability Assessment:** Uses large language models such as **Mixtral and Llama** to evaluate whether each test step is **clear, specific, and actionable**. Steps classified as vague or incomplete are flagged for refinement.

Since test steps are extracted from multiple sources, inconsistencies are inevitable. By systematically detecting and flagging these issues, the system ensures that engineers receive a refined, high-quality set of test steps without having to manually compare documents.

By automating this validation step, the system reduces misinterpretations, enhances test case reliability, and ensures that all necessary steps are precise and executable. This structured approach significantly accelerates test case development while improving accuracy and efficiency.

6 Coordinate

Even after test steps are compiled and validated, engineers often need to clarify missing details or ambiguous instructions. Without a structured system, this requires manually tracking down the original authors or contributors of test cases, leading to unnecessary delays. The system streamlines this process by automatically extracting and associating relevant metadata with each test step.

The system will analyze the Excel file properties and embedded metadata to retrieve useful details such as:

- **Author Information:** Identifies the original creator and last editor of the sheet, allowing engineers to contact the relevant contributor directly.

- **Modification History:** Extracts timestamps and past changes to determine when and by whom a test step was last updated.

- **Relevant Teams or Departments:** Where applicable, the system associates test cases with specific engineering groups to streamline communication.

By presenting this metadata alongside the compiled test case list, the system significantly reduces the time engineers spend searching for clarification. Instead of manually investigating spreadsheet ownership, they can immediately access contact details for the most relevant person.

Additionally, this coordination process enhances collaboration by creating a more transparent workflow. Engineers can quickly identify who is responsible for specific test cases, ensuring faster resolution of ambiguities and a more efficient development cycle.

7 Clean

After test steps have been compiled, compared, and coordinated with relevant contacts, the final stage ensures that the output is fully refined, structured, and ready for engineer review. While the initial cleaning focused on preprocessing raw Excel data, this final cleaning step eliminates any lingering inconsistencies, redundant information, or unclear test steps that may have survived earlier stages.

The processed test case data will be presented through a **clear API call** that integrates seamlessly with a **Streamlit or Dash online app**, allowing TOSCA engineers to access, filter, and interact with test cases in an intuitive interface. Instead of navigating through raw spreadsheets, engineers will be able to:

- **Search and Retrieve Test Cases:** Engineers can input queries to pull relevant test cases based on keywords, past modifications, or metadata.

- **View Structured Test Steps:** The app will display a fully refined test case list, ensuring all steps are clear, specific, and actionable.

- **Access Contact Information:** Metadata linking test cases to relevant contributors will be accessible, allowing engineers to quickly resolve ambiguities.

- **Export Data for Further Use:** The system will support exporting test cases in structured formats such as JSON or CSV for integration into other workflows.

The API and online interface will ensure that the final cleaned dataset is not just well-structured but also highly accessible. This approach eliminates the inefficiencies of manual spreadsheet searches, providing engineers with an interactive, real-time solution for managing test cases.

8 Conclusion

The development of this automated test case refinement system has successfully addressed the inefficiencies faced by TOSCA engineers in managing and structuring test steps. Through the structured workflow of **Clean, Compile, Compare, Coordinate, and Clean**, the system eliminates redundant steps, detects inconsistencies, ensures specificity in test descriptions, and provides engineers with a streamlined API-based interface for managing test cases.

Each stage of the process has been completed, including the implementation of data cleaning, test case compilation, duplicate detection, metadata extraction, and the final structured presentation through an online interface. The only remaining task is to stitch these components together into a cohesive pipeline, ensuring smooth integration across all steps.

The final implementation is expected to be fully operational following our meet-up in Orlando next week. With this system in place, engineers will benefit from a significant reduction in manual effort, increased test case reliability, and a more efficient workflow for managing test documentation.