# Product Requirements Document (PRD)

## AI-Powered Test Case Processing API

|  |  |  |  |
| --- | --- | --- | --- |
| **Document Version** | **Author** | **Date** | **Status** |
| 1.0 | Gemini Assistant | October 1, 2025 | Draft |

### 1. Overview & Vision

This document outlines the requirements for an internal API designed to automate the tedious process of formatting software test cases. By accepting a .csv file, the service will leverage the **Qwen Large Language Model** to intelligently parse unstructured test step descriptions into a clean, structured JSON format, preparing the data for import into test management systems or databases.

### 2. The Problem

Quality Assurance (QA) teams often write test cases in spreadsheets for convenience. However, the "Test Step" descriptions are frequently written in a single cell with numbered lists or bullet points. Before these test cases can be used in automated systems or formal documentation, they must be manually separated and reformatted, which is a time-consuming, repetitive, and error-prone task.

### 3. Goals & Objectives

* **Primary Goal:** To eliminate the manual effort required to reformat test case steps from CSV files.
* **Key Objectives:**
  + Achieve >95% accuracy in correctly identifying and separating test steps from a single text block.
  + Reduce the time to format a 100-row test case file from over an hour to under 2 minutes.
  + Provide a simple, developer-friendly API endpoint that can be easily integrated into other workflows.

### 4. Target Audience

* **QA Engineers & Test Analysts:** The primary users who will benefit from the automation.
* **Software Developers & DevOps Engineers:** Who may integrate this API into CI/CD pipelines or data migration scripts.

### 5. Functional Requirements (Features)

#### 5.1. API Endpoint

* **Endpoint:** POST /api/v1/process-test-cases
* **Request:** multipart/form-data
  + **file**: The .csv file containing the test cases.
  + **target\_columns (Optional)**: A JSON string specifying the column names to use. E.g., {"title": "Scenario Description", "steps": "Test Step", "result": "Expected Results"}. If not provided, the AI will attempt to identify them automatically.
* **Response (Success - 200 OK):** A JSON object containing a list of processed test cases.  
  {  
   "status": "success",  
   "processed\_count": 3,  
   "results": [  
   {  
   "data": {  
   "test\_title": "To verify if the dashboard page loads successfully upon login.",  
   "test\_steps": [  
   "Open the '[https://app.nostro.co/sign-in](https://app.nostro.co/sign-in)' URL.",  
   "Observe the Homepage loads successfully."  
   ],  
   "expected\_result": "The dashboard page should load successfully, displaying all widgets and data."  
   }  
   },  
   // ... more results  
   ]  
  }
* **Response (Error - 4xx/5xx):**  
  {  
   "status": "error",  
   "message": "A descriptive error message (e.g., 'File not provided', 'Could not identify test step column', 'LLM API is unavailable')."  
  }

#### 5.2. CSV Processing Logic

1. The system shall accept .csv files up to a configurable limit (e.g., 5MB).
2. It will read the CSV and identify its header row.

#### 5.3. AI-Powered Column Identification

1. If the target\_columns payload is **not** provided, the system will send the list of header names to the Qwen model.
2. The prompt will ask the LLM to identify which columns correspond to the test title, the test steps, and the expected result.
3. The system must be able to handle cases where the LLM cannot confidently determine the columns and return a helpful error.

#### 5.4. AI-Powered Test Step Extraction

1. For each row in the CSV, the system will extract the text from the identified "test steps" column.
2. This text will be sent to the Qwen model with a specific prompt.
   * **Example Prompt:** "You are a text processing expert. Parse the following text block, which contains multiple test steps. Extract each individual step. Your output must be a valid JSON array of strings, with each string being one step. Do not include the original numbers or bullets. Text: '1. Open the URL.\n2. Enter invalid link.\n3. Click Go back button.'"
3. The system will parse the JSON array returned by the LLM.

#### 5.5. JSON Transformation

1. The system will construct the final JSON object for each row, using data from the respective columns for test\_title and expected\_result, and the list of strings from the LLM for test\_steps.

### 6. Technical Stack

* **Programming Language:** Python 3.9+
* **API Framework:** FastAPI or Flask
* **Data Handling:** Pandas
* **LLM Integration:** Direct API calls to a hosted Qwen model endpoint.
* **Deployment:** Docker container

### 7. Assumptions

* The input .csv files are well-formed and UTF-8 encoded.
* The test steps within a single cell are typically delimited by newlines and/or numbering (e.g., 1., 2., a., b., -).
* Access to a reliable, hosted Qwen model API is available.

### 8. Out of Scope for v1.0

* A graphical user interface (GUI) for file uploads.
* User authentication or account management.
* Storing processed results in a database.
* Real-time processing via WebSockets.