# Project Roadmap: AI-Powered Test Case Processing API

This roadmap outlines the phased development of an API that ingests CSV files containing software test cases and uses the Qwen LLM to parse and structure them into a standardized JSON format.

### Phase 1: Core Functionality (Minimum Viable Product - MVP)

**Goal:** Build a functional, proof-of-concept API that can process a single, well-formatted CSV file and return the correct JSON output.

* **Q1 - Weeks 1-2: Setup & API Foundation**
  + [ ] **Environment Setup:** Set up a Python development environment (e.g., using venv or Docker).
  + [ ] **Framework Selection:** Choose a lightweight API framework (FastAPI is recommended for its speed and automatic documentation).
  + [ ] **Basic Endpoint:** Create a single /process-csv POST endpoint that accepts a file upload.
  + [ ] **Qwen Model Integration:** Set up the API client to connect to the Qwen model's endpoint. Securely manage API keys.
* **Q1 - Weeks 3-4: CSV Parsing & LLM Logic**
  + [ ] **CSV Reading:** Implement logic to read the uploaded CSV file into memory (e.g., using the pandas library).
  + [ ] **Hardcoded Column Mapping:** Initially, assume fixed column names like Test Step, Scenario Description, and Expected Results.
  + [ ] **Core LLM Prompting:** For each row, send the content of the Test Step cell to the Qwen model with a carefully crafted prompt to extract the steps into a list.
  + [ ] **JSON Transformation:** Write the script to assemble the final JSON structure for each row based on the LLM's output and the other columns.
  + [ ] **Unit Testing:** Write basic tests for file upload, CSV parsing, and the JSON transformation logic.

### Phase 2: Intelligence and Robustness

**Goal:** Make the API smarter by removing hardcoded assumptions and improving its reliability.

* **Q2 - Weeks 5-6: Dynamic Column Identification**
  + [ ] **Header Analysis:** Implement the AI-driven column identification. When a CSV is uploaded, send its header row to Qwen with a prompt like: *"From this list of column names, identify the one that contains test steps, the one for the test title, and the one for the expected result. Return the names in a JSON format."*
  + [ ] **Fallback Mechanism:** If the LLM fails to identify the columns confidently, return a specific error asking the user to name their columns standardly (e.g., 'Test Step').
  + [ ] **Enhanced Error Handling:** Implement comprehensive error handling for invalid file types, malformed CSVs, and API-level errors from the LLM.
* **Q2 - Weeks 7-8: Performance and Usability**
  + [ ] **Batch Processing:** Allow the API to process all rows in the CSV in a single request and return a list of JSON objects.
  + [ ] **Asynchronous Processing:** For very large files, convert the endpoint to be asynchronous. The user uploads the file and gets a job ID, which they can use to poll for the result later.
  + [ ] **Logging:** Implement structured logging to monitor API requests, successes, and failures.

### Phase 3: Production Readiness & Scaling

**Goal:** Prepare the API for real-world usage with a focus on security, scalability, and maintainability.

* **Q3 - Weeks 9-10: Security & Deployment**
  + [ ] **Authentication:** Implement API key-based authentication to secure the endpoint.
  + [ ] **Input Validation:** Add strict validation for file size and content type.
  + [ ] **CI/CD Pipeline:** Set up a continuous integration/continuous deployment pipeline for automated testing and deployment.
  + [ ] **Containerization:** Finalize the Docker configuration for easy deployment to cloud services (e.g., AWS, Google Cloud, Azure).
* **Q3 - Weeks 11-12: Monitoring & Optimization**
  + [ ] **Health Check Endpoint:** Add a /health endpoint for monitoring the API's status.
  + [ ] **Performance Monitoring:** Integrate tools to monitor API latency, error rates, and resource usage.
  + [ ] **Prompt Optimization:** Continuously refine the prompts sent to the Qwen model to improve accuracy and reduce token usage (cost).