Automated DVP Generation using FLAN-T5 and Prompt Engineering

# 1. Introduction

In this project, an automated testing framework is built to extract vehicle system requirements from a document and generate corresponding Design Verification Plan (DVP) test cases. The goal is to produce fields such as Test Description, Precondition, Action, and BCM Expected Output automatically using a Large Language Model (LLM).

# 2. Model Used - FLAN-T5

FLAN-T5 is a fine-tuned version of Google's T5 (Text-To-Text Transfer Transformer), designed specifically for instruction-following tasks. It is trained on a mixture of open-ended prompts and question-answering tasks. It allows converting various tasks into a text generation problem, making it suitable for requirement understanding and test case generation.

# 3. Technologies Used

- FLAN-T5 model (from Hugging Face)

- Transformers library by Hugging Face

- PyTorch for running model inference

- Python and Jupyter Notebook for development

- BeautifulSoup for parsing the .docx or HTML input

- Pandas and Openpyxl for tabular data output

# 4. Prompt Engineering

Prompting was key to generating meaningful and accurate outputs. Initially, the user defined how the BCM expected output should appear by showing a few sample rows from an existing DVP Excel sheet. Prompts were designed to mimic those outputs.

Example Prompt:  
Given the automotive requirement: 'The system shall activate the wiper when it rains.'  
Generate the following fields in structured form:  
Test Description: ...  
Precondition: ...  
Action: ...  
BCM Expected Output: ...

# 5. Input Format

The requirements were stored in a Word file (`Project.docx`). These requirements were not well-structured, so they were parsed using Python and BeautifulSoup to extract meaningful lines.

# 6. Output Format

The output was a structured Excel file with the following columns:  
- Requirement ID  
- Test Applicability  
- Test Case ID  
- Test Description  
- Precondition  
- Action  
- BCM Expected Output

# 7. Workflow Summary

1. Read and parse unstructured requirements from a Word (.docx) file  
2. For each requirement, extract the text  
3. Send carefully designed prompts to FLAN-T5 model  
4. Parse the model response into structured columns  
5. Display and optionally export the data as an Excel sheet

# 8. Improvements & Future Scope

- Integrate RAG (Retrieval-Augmented Generation) to reduce hallucination

- Add validation rules for verifying output correctness

- Use fine-tuned models trained specifically on DVPs and BCM datasets

- Enable GUI for non-technical users to interact with the model