
📄 **Project Report: Vehicle Gherkin Scenario Generator**

🔧 **Project Title: Vehicle Testing Scenario to Gherkin Step Converter**

📘 Overview

This project is a **Streamlit-based web application** designed to convert **free-form vehicle testing scenarios** into **Gherkin syntax** using predefined templates and **Google Gemini 1.5** large language model (LLM). By combining **prompt engineering** with **retrieval-augmented generation (RAG)**, the app ensures that the output remains consistent with industry-standard Gherkin formatting.

🎯 Objective

To assist vehicle engineers, testers, and automation teams by automating the generation of BDD (Behavior-Driven Development) test cases from natural language descriptions. This reduces manual effort, increases standardization, and accelerates test development workflows.

🧩 Key Features

- 🧠 **LLM Integration**: Uses Google's Gemini 1.5 Flash model to interpret complex scenario text.
- 📚 **Template-Based RAG**: Uses a FAISS-powered vector store to retrieve domain-specific Gherkin templates.
- 💬 **Dynamic Prompting**: Combines user scenario input with template context for accurate generation.
- 🖥️ **Streamlit UI**: Simple interface to input scenarios and view Gherkin output instantly.

🧱 System Components

1. Gherkin Templates File

The file gherkin_templates.txt contains structured, parameterized examples of Gherkin syntax. These include Given, When, and Then steps typical in vehicle systems testing. Examples:

Given the vehicle is in <vehicle_state>

When the driver performs <driver_action>

Then the system shall trigger <alert_type> alert

These templates cover:

- **Vehicle State and Inputs** (Given)
- **Sensor and Driver Events** (When)
- **System Responses and Logs** (Then)

2. Application Code: app.py

a. Gemini Model Configuration

```
genai.configure(api_key=os.getenv("GOOGLE_API_KEY"))

model = genai.GenerativeModel("gemini-1.5-flash")
```

Connects to the Gemini API using the environment key.

b. Text Embedding and Indexing

```
embeddings = GoogleGenerativeAIEmbeddings(model="models/embedding-001", ...)

db = FAISS.from_documents(texts, embeddings)
```

- Splits the Gherkin templates into smaller chunks.
- Embeds them and stores in a FAISS vector store.
- Returns a retriever to support template retrieval for prompt injection.

c. Gherkin Generation via Prompt

```
def generate_gherkin_scenario(user_input, template_context)
```

- Combines user scenario + Gherkin templates in a structured prompt.
- Passes it to Gemini for structured Gherkin output.

d. Streamlit UI

st.text_area("Vehicle Testing Scenario", height=200)

- Accepts user scenario as input.
- Displays LLM-generated Gherkin result on button click.

🛠️ Usage Instructions

1. **Environment Setup**
    - Install dependencies:
    - pip install streamlit langchain faiss-cpu google-generativeai langchain-google-genai
    - Set up the GOOGLE_API_KEY in your environment.
2. **Run the App**
3. streamlit run app.py
4. **User Input**
    - Example:
    - The vehicle is driving in rain and the lane-keeping assist is enabled.
    - Output:
    - Feature: Lane-Keeping in Rain
    - Scenario: Lane keeping in rainy weather
    - Given the vehicle is in driving state
    - And the environment condition is rain
    - And the lane-keeping assist feature is enabled
    - Then the system shall maintain lane position

📈 Benefits

- **Automation**: Reduces manual effort in Gherkin creation.
- **Consistency**: Enforces structured test definitions across teams.
- **Scalability**: Supports various scenarios across vehicle domains.

- **Adaptability**: Can expand with more templates or scenario domains (e.g., ADAS, EV systems).

🚧 Limitations & Future Enhancements

Limitation	Suggested Improvement
No multi-turn interaction	Add session memory to refine scenarios iteratively
Output relies on prompt design	Implement prompt chaining with deeper context
Only uses a static .txt file	Replace with dynamic template database
No validation of generated Gherkin	Add post-checks or syntactic validation

---

📁 File Structure

app.py

gherkin_templates.txt

.env (optional for GOOGLE_API_KEY)

✅ Conclusion

The **Vehicle Gherkin Scenario Generator** bridges the gap between domain expertise and test automation by translating vehicle testing scenarios into structured Gherkin format using the power of LLMs and prompt templating. It's a helpful tool for streamlining the development of behavior-driven testing strategies in automotive software.
