Proposal Document for Automated Test Case Generation and Verification System with UI

# Objective:

The goal of this project is to automate the process of generating, mapping, and verifying software test cases based on Software Requirements Specification (SRS) documents. By utilizing LangChain agents and large language models (LLMs), this system will ensure that software requirements are adequately covered by corresponding test cases, while also generating new test cases where needed. The system will integrate a user interface (UI) to enhance user interaction, allowing users to upload SRS documents and view the extracted requirements, mapped test cases, and the relationships between them.

# Technologies Used:

1. LangChain Framework: Provides structure to define and organize agents that perform complex tasks, such as generating and verifying test cases.

2. Google Generative AI (Gemini Model): Utilized for generating test cases and processing text from the SRS document.

3. Hugging Face Embedding Models: Embedding model: sentence-transformers/all-mpnet-base-v2.

4. FAISS (Facebook AI Similarity Search): A vector store for searching and retrieving similar embeddings.

5. PyPDFLoader: Used to extract text from SRS documents in PDF format.

6. Frontend: HTML, JS, CSS with Flask templates

7.Backend: Flask.

# System Architecture:

Step 1: Load and Extract Text from SRS DocumentAgent: SRSLoaderAgent, Technology: PyPDFLoader

Step 2: Extract Requirements from SRS DocumentAgent: RequirementsExtractorAgent, Technology: LLMChain, Google Gemini Model

Step 3: Generate and Verify Test CasesAgent: TestCaseGeneratorAgent, Technology: LLMChain, Google Gemini Model

Step 4: Map Requirements to Test CasesAgent: TestCaseMappingAgent, Technology: Hugging Face Embeddings, FAISS Vector Store

Step 5: Human-in-the-Loop Validation (UI Feature) Technology: Flask. Users will be able to validate the mappings through a UI.

# Detailed Procedure:

1. SRS Document Upload

2. SRS Document Loading and Extraction

3. Requirements Extraction

4. Test Case Generation

5. Test Case Mapping and Verification

6. UI-Driven Human-in-the-Loop Validation

7. Final Output and Reporting

# Applications and Use Cases:

1. Automated Software Testing

2. Quality Assurance (QA)

3. Agile Development

4. DevOps Integration

5. Collaborative Review

# Advantages:

1. Efficiency

2. Scalability

3. Accuracy

4. Human Validation

5. User-Friendly

# Conclusion:

The proposed system offers a robust solution for automating the generation and validation of test cases based on SRS documents, utilizing LLMs, embeddings, and human feedback. By incorporating an interactive UI, we enhance collaboration and transparency, allowing users to ensure the accuracy of generated test cases and their mapping to requirements. This system will streamline the software testing process, improve coverage, and integrate seamlessly into existing workflows.