**Project Report: Automated Test Case Generation from Requirements**

**Objective:** Develop a tool to automatically generate test cases from software requirements written in natural language. The goal is to automate the process of converting requirements into structured test cases to streamline testing and improve coverage.

**Process Overview:**

1. **Data Loading and Cleaning:**
   * Loaded requirements from a CSV file.
   * Cleaned the text by converting it to lowercase and removing special characters and extra spaces to standardize the input.
2. **Text Preprocessing:**
   * **Tokenization:** Split the cleaned text into individual words using NLTK’s word\_tokenize.
   * **Lemmatization:** Reduced words to their base forms using NLTK’s WordNetLemmatizer to ensure consistency in text analysis.
3. **Entity Extraction:**
   * **Named Entity Recognition (NER):** Applied spaCy to identify named entities in the text, such as people, organizations, and locations.
   * **Part-of-Speech (POS) Tagging:** Used NLTK to classify words into their grammatical categories (nouns, verbs, adjectives, etc.).
   * **Specific Entity Extraction:** Extracted key components from POS tags, including actors, actions, conditions, and outcomes.
4. **Test Case Generation:**
   * Developed various templates to generate test cases based on the extracted entities. Templates included:
     + **Basic Functionality:** Verify basic actions and interactions.
     + **Condition-Based:** Check functionality under specific conditions.
     + **Negative Cases:** Ensure certain actions do not occur under specific conditions.
     + **Outcome-Based:** Validate expected outcomes from actions.
     + **Multiple Actors:** Test interactions between multiple actors.
     + **Sequential Actions:** Validate the sequence of actions.
     + **Conditional Outcomes:** Ensure specific outcomes occur under given conditions.
     + **Performance Checks:** Measure performance metrics during actions.
5. **Output and Export:**
   * Generated test cases for each requirement based on the templates.
   * Compiled test cases into a structured DataFrame.

**Results:**

* Automated the transformation of natural language requirements into structured test cases.
* Provided a variety of test case templates to cover different testing scenarios.