Use Case Development for Unit Testing

Project: AI-Based Spellchecker (TextBlob + Google Colab)

Team/Members: MEGAN PAIGE MULHOLLAND
Connor Flok, Megan Mulholland, Jake Hauff, Ayo Abraham

Team Member Contribution:
Connor Flok (Use case 1), Megan Mulholland (Proofread/final checks, Use case 4), Jake Hauff (Use Case 2), Ayo Abraham (Use case 3,5). We also discussed as a group and brainstormed together.

Use Case 1: Spellcheck Single Misspelled Word

Purpose: To verify that the spellchecker accurately identifies and corrects a simple misspelled word.

Preconditions:

- TextBlob library is properly installed and imported.
- The spellcheck_text() function is available and operational.

Input Data: "helo" or any misspelled word

Expected Output: Returns "hello" as the corrected output. (or the corrected word)

Edge Case:
Input: "helolllll" → Expected output: "hello" (closest valid correction without failure).

Use Case 2: Spellcheck Full Sentence with Multiple Errors

Purpose: To ensure the spellchecker can process and correct multiple misspelled words within a sentence.

Preconditions:

- Function accepts full sentences or paragraphs.
- Google Colab notebook session is active.
- Use Case 1 is successful.

Input Data: “Ysterday it was suny and wam out.”

Expected Output: “Yesterday it was sunny and warm out.”

Edge Case:
Input: “Ysterday   it was suny and wam out.” Result: Expected to handle extra spaces gracefully and still produce a properly spaced corrected output.

Use Case 3: Display Suggested Corrections

Purpose: To validate that the system provides a list of possible corrections for each detected misspelled word.

Preconditions:

- Suggestion function (e.g., get_suggestions(word)) is implemented and functional.
- User interface or console output displays suggestions clearly.

Input Data: "speling"

Expected Output: Suggestions: [“spelling”, "spieling", "spewing"]

Edge Case:
Input: "hte" → Should return suggestions like ["the", "hate", "hit"] without crashing or errors.

Use Case 4: Handle Empty or Invalid Input

Purpose: To ensure the system handles empty or invalid text input without errors.

Preconditions: Error handling and input validation are implemented.

Input Data: "" (empty string) or " " (spaces only)

Expected Output: Message displayed: "Please enter text to spell check."

Edge Case:
Input: "\n\n" (newline characters only) → Expected same message; program should not crash or hang.

Use Case 5: Preserve Punctuation and Capitalization

Purpose: To confirm that punctuation and capitalization are maintained after correction.

Preconditions: Spellcheck function processes punctuation marks and capitalization.

Input Data: "I realy like Pyhton!"

Expected Output: "I really like Python!" (corrected words, punctuation preserved, capitalization maintained)

Edge Case:
Input: "HELLO wrld!" → Expected Output: "HELLO world!" (preserves capitalization style and punctuationm but also fixes the misspelled word).