**Technical Design Document Template**

**Name:** Ethan Back

**Date Created:** 9/21/25

**Program Description:**

This program analyzes an input email message to determine its likelihood of being spam. It scans the message for a predefined list of common spam keywords and phrases. For each detected keyword or phrase, it adds points to a spam score. Based on the total score, the program rates the email's spam likelihood and displays which spam indicators were found. The program helps users quickly assess whether an email might be suspicious or junk.

Alot of “cleaning” is done to remove things like excess spacing, lines, and punctuation from the text. All of which caused errors to come up while writing the code and running testcases, specifically the excess lines that came from copy pasting the test case.

**Functions used in the Program (list in order as they are called):**

1. **Function Name:** normalize\_spaces

**Description:** Removes extra spaces from a string, reducing multiple spaces to a single space, and removing leading/trailing spaces

**Parameters:**

* text(string): the input string you want to normalize the spaces in

**Variables:** None

**Logical Steps:**

* Uses a regex substitution to replace all sequences of whitespace characters with a single space

**Returns:**

* (string) The input text with normalized spacing

2. **Function Name:** remove\_punctuation

**Description:** Removes all punctuation characters from a string.

**Parameters:** text (string): The input string from which to remove punctuation.

**Variables:** None

**Logical Steps:** Uses Python’s str.translate with a translation table that deletes all punctuation characters as defined by string.punctuation.

**Returns:** (string) The input text stripped of punctuation characters

3. **Function Name:** clean

**Description:** Cleans input text by lowercasing, normalizing spaces, and removing punctuation to standardize the text for analysis.

**Parameters:** text (string): The input text to clean.

**Variables:**

* rinse\_1 (string): Result after normalizing spaces and lowercasing.
* rinse\_2 (string): Result after removing punctuation from rinse\_1.

4. **Function Name:** spam\_tracker

**Description:** Scans a message for a list of spam keywords and phrases, calculating a spam score and tracking which keywords were found.

**Parameters:** message (string): The email message text to scan.

keywords (list of strings): List of spam keywords and phrases to check for.

**Variables:**

* cleaned\_message (string): The cleaned message text.
* words (list of strings): The cleaned message split into individual words.
* word\_set (set of strings): Set version of words for fast lookup.
* score (int): Counts the total number of detected spam keywords/phrases.
* matched\_words (list of strings): Holds detected keywords/phrases for reporting.
* cleaned\_keyword (string): Each keyword cleaned before matching to ensure consistent comparison.

**Logical Steps:**

* Clean the message for consistent comparison.
* Split the message into words and create a set for quick lookup.
* For each keyword:
* Clean the keyword similarly.
* If keyword contains spaces (a phrase), check if it exists as a substring in the cleaned message.
* Else (single word), check if it exists in the set of words.
* If found, increment score and add original keyword to matched words.

**Returns:**

* score (int): The total spam score.
* matched\_words (list): List of spam keywords/phrases found in the message.

**Logical Steps:**

* Convert text to lowercase.
* Normalize spaces with normalize\_spaces().
* Remove punctuation with remove\_punctuation().

**Returns:** (string) The cleaned and normalized text.

5. **Function Name:** spam\_rating

**Description:** Converts a numeric spam score to a qualitative rating. \*Assigns meaning to certain value brackets\*

**Parameters:**

* score (int): Spam score to rate.

**Variables:** None

**Logical Steps:**

* Use conditional checks on score ranges to assign ratings:
* 0: Very Low
* 1-3: Low
* 4-6: Moderate
* 7-10: Likely
* 10 or higher: Very Likely

**Returns:**

(string) The qualitative spam likelihood rating.

6. **Function Name:** main

**Description:** Manages user input, processes the email message for spam detection, and displays results.

**Parameters:** None

**Variables:**

* keywords (list of strings): Predefined spam keywords and phrases.
* message\_lines (list of strings): Lines of user input forming the email message.
* empty\_line\_count (int): Counter to detect two consecutive blank lines signaling end of input.
* message (string): Full email message joined from input lines.
* score (int): Spam score from spam\_tracker().
* matches (list): Detected spam keywords/phrases.
* likely\_hood (string): Spam likelihood rating from spam\_rating().

**Logical Steps:**

* Print instructions for user input.
* Read user input lines until two consecutive empty lines are detected.
* Join input lines into one message string.
* Call spam\_tracker to get spam score and matches.
* Call spam\_rating to get qualitative rating.
* Print spam score, likelihood, and matched spam indicators (if any).

**Returns:** None

**Logical Steps:**

1. User runs program, main() is called.
2. main() prompts user to input an email message (supports multi-line input).
3. User inputs message, ending with two consecutive blank lines.
4. main() calls spam\_tracker() with the message and spam keywords list.
5. spam\_tracker() cleans the message and keywords, scans for keywords and phrases, returning a score and matched words.
6. main() calls spam\_rating() with the score to get a qualitative likelihood.
7. main() prints the spam score, likelihood, and the matched keywords/phrases (if any).
8. Program ends.

**Link to your repository:** <https://github.com/if-it-Works-it-Works/COP2373>

**Output Screenshot: (make sure big enough so I can see)**

