**Technical Design Document Template**

**Name:** Alanys Suazo

**Date Created:** 09/21/2025

**Program Description:**

The purpose of this program is to scan user input as a string, after which that string will be scanned for key phrases seen and used in scam and spam emails. It should return the number of substrings and set an accumulator, which will tell the user the likelihood of an email being a scam.

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

1. **Function Name:** main

**Description:** Controls the overall program flow, including user input, spam analysis, and repeated execution.

**Parameters:** n/a

**Variables:**

* Keywords (list of str): Contains all spam-related phrases used for detection.
* message(str): The email message entered by the user.
* score (int): The spam score returned by spam\_sorter
* matched\_keywords (list of tuples): Keywords found in the message and their counts.
* Rating (str): Spam likelihood returned by scam\_score\_counter
* Again (str): User input to determine whether to continue or exit the loop.

**Logical Steps:**

1. Display program title and pause briefly.

2. Start a loop to allow repeated message analysis.

3. Prompt user to enter an email message.

4. Pass the message and keyword list to the spam\_sorter

5. Pass the score to the scam\_score\_counter to get a rating.

6. Display the score, rating, and matched keywords.

7. Ask the user if they want to analyze another message.

8. Exit the loop if the user types anything other than "yes" or "y".

**Returns:** n/a

2. **Function Name:** spam\_sorter

**Description:** Scans the cleaned message for known spam keywords and calculates a spam score based on how many keywords are found.

**Parameters:** message and keywords

**Variables:**

* spam\_score *(int)*: Accumulator that tracks the total number of keyword matches.
* matched *(list of tuples)*: Stores each matched keyword and its count.
* word *(str)*: Each keyword being checked.
* count *(int)*: Number of times a keyword appears in the message.

**Logical Steps:**

1. Clean the message using .

2. Loop through each keyword in the list.

3. Check if the keyword exists in the message.

4. Count how many times it appears.

5. Add the count to and store the keyword and count in .

**Returns:** Total number of keyword matches. List of tuples showing which keywords were found and how often.

3. **Function Name:** clean\_message

**Description:** Prepares the input message for keyword scanning by converting it to lowercase, removing punctuation, and collapsing extra spaces. (It may be over the top, but I didn’t know how else to fix my bugs; I later realized the reason for the bug was something I hadn’t taken into account but left this function in because you never know.)

**Parameters:** message

char (str): Each punctuation character to be removed from the message.

1. Convert the message to lowercase.
2. Loop through a list of punctuation characters and remove each one from the message.
3. Replace multiple spaces with a single space using .split() and ' '.join().

**Returns:** A cleaned version of the message *(str)*, ready for keyword scanning.

4. **Function Name:** spam\_score\_counter

**Description:** Interprets the spam score and returns a human-readable rating of how likely the message is spam.

**Parameters:** score

**Variables:** n/a

**Logical Steps:**

1. If score is 0, return a message indicating it's not spam.
2. If score is between 1 and 3, return a low-risk warning.
3. If score is between 4 and 6, return a medium-risk warning.
4. If score is 7 or higher, return a high-risk warning with a humorous message.

**Returns:** A string describing the likelihood of the message being spam.

**Logical Steps:**

I'm feeling rushed so here you go:

main -> spam\_sorter -> clean\_message -> scam\_score\_counter -> back to main

**Link to your repository:** [Alanys-SG/COP2373 at master](https://github.com/Alanys-SG/COP2373/tree/master)

**Output Screenshot: A screenshot of a computer

AI-generated content may be incorrect.**