**Technical Design Doc**

Nick Herrera

Sep 24, 2025

**Description:**

This program analyzes a user-input email to detect a presence of common spam words and phrases. Its purpose is to scan the message’s content for 30 predefined spam indicators as well as provide a spam score along with a spam likelihood rating. This helps determine how likely it is that the message is spam, based on the number of flagged words or phrases.

**Functions used:**

1. Func Name – main()
2. Description - Handles user input, and calls for both other functions, and prints the results.
3. Parameters – 0
4. Variables - email\_message(string) – stores the message entered by user

Score(int) – the number of spam words found in the message

matched\_words: (list of strings) spam words or phrases found in email.

rating(string) – likelihood of the spam being returned by rate\_spam

**Logical Steps:**

1. Prompt user for email message.
2. Stores user input.
3. Call analyze\_email() to get spam score and matched words.
4. Call rate\_spam() to determine how likely the message is spam.
5. Print the spam score and rating.
6. Display which spam words were found (if there are any).

Returns: none

2. Func name – analyze\_email

Description – scans and counts the # of spam words and phrases.

Parameters – text(string) – email being scanned.

Variables - spam\_score (int): counts the number of spam words found.

matched (list of strings): stores the spam keywords.

text (string): lowercase version of input message for comparison.

Logical Steps:

1. Convert input text to lowercase for case-insensitive comparison.
2. Loop through each spam keyword in the list SPAM\_WORDS.
3. If the keyword is in the message, increase spam\_score and add the word to matched.

**Returns**: spam\_score (int), and matched.

1. Function Name: rate\_spam

Description: based on the spam score, returns a string describing the likelihood that the message is spam.

Parameters: score (int): the number of spam keywords found.

Variables: none

**Logical Steps:**

1. If score is 0, return "Not spam".
2. If score is 1–3, return "Unlikely spam".
3. If score is 4–6, return "Possible spam".
4. If score is 7–10, return "Likely spam".
5. If score is over 10, return "Very likely spam".

**Returns:** string such as "Not spam", "Likely spam", etc.

**Logical Steps:**

1. The program starts by calling main() function.
2. Inside main, the user is prompted to enter a message.
3. analyze\_email(email\_message) is called to get the spam score and the matched keywords.
4. rate\_spam(score) is called to convert the score into the spam likelihood description.
5. The program prints the spam score, rating, and matched keywords (if any).

Repository link - <https://github.com/nherrer2/COP2373>

Screenshot - 