**Text Processing Plan**

**Requirements**

1. Reading and Processing the Data
   1. This code should be able to read the spam or not spam dataset form a CSV file, process each email in the dataset to create relevant features, and features to compute word count, bigram counts, and work stats.
2. Email Representation
   1. This code should have a class to represent each email with its features in which can be stored into a list of name values.
3. Saving the features and summaries
   1. This code should save the features from each email into a CSV file and then the summary should be saved to a separate one
4. Displaying the features and summaries
   1. This code should display the features of the statistics such as what is the size of each word or how many times that word appeared.
5. Distance Computation
   1. This code should use a method in which computes the distance between the emails based on the certain features of each
6. Classification
   1. This code with take all the features and classify the email as spam or not spam

**UML**

1. emailId: int
2. features: dict
3. isSpam: bool
4. computeDistance(otherEmail: email)
5. computeDistanceFromSum(summary: dict)
6. dataset :List[Email]
7. summary: dict
8. readDataFromCsv(filePath: str)
9. processEmails()
10. saveSumToCsv
11. displayEmailFeatures(emailID; int)
12. displaySum()

**Test**

1. Load a sample CSV file with email data.
2. Test reading and processing of the dataset.
3. Calculate features for a few emails and verify the correctness of feature computation.
4. Save the computed features and summary to CSV files and check the saved data.
5. Test distance computation between two emails using various distance metrics.
6. Test distance computation between an email and the summary data.
7. Load pre-trained models for spam and not spam.
8. Classify emails as spam or not spam based on distances to models.
9. Verify that the program can correctly classify emails.