Draft README file

‘’’ The project scope is detailed under the project scope and guidance.doc saved under the 1. Requirements and useful project info folder.

Below is a summary of the

1. project scope and the main features to address the requested outcome
2. the stretched goals that this project is trying to address, up to a point, as these are very complex goals that go beyond my current programming experience
3. approach and logic used for providing the required output

‘’’

# DNA Identification Service

## Overview

The DNA Identification Service is designed to process DNA sequences, identify the closest match from a provided database, and output the closest sequence along with its similarity score. The project extends into stretch goals that involve calculating probabilities and building a phylogenetic tree based on the sequence data.

### Key Features:

- \*\*Identify the closest sequence\*\*: Given a query sequence, the program will compare it against a database of sequences and find the closest match based on sequence similarity.

- \*\*Output\*\*: The output will include the closest matching sequence's identifier and the score indicating how closely it matches the input sequence.

### Stretch Goals:

1. \*\*Probabilities\*\*: Compute the probabilities of the query sequence matching different sequences in the database, and calculate p-values to assess the significance of the matches.

2. \*\*Phylogenetic Tree\*\*: Construct a reconstructed phylogeny based on the sequence alignment results, showing how the sequences are related.

### Approach and Logic summary:

The program will follow these steps to derive the solution:

1. Input Validation: Ensure that the input sequence file (`mystery.fa`) and the sequence database (`dog\_breeds.fa`) are in the correct format (FASTA).

2. Sequence Alignment: Use the `pairwise2` module to align the test sequence against each sequence in the database, using a global alignment strategy.

3. Match Scoring: After performing the alignment, the program will extract the alignment score, which indicates how closely two sequences match.

4. Closest Sequence Identification: The program will find the sequence in the database with the highest alignment score, indicating the closest match to the input sequence.

5. Output Results: Display and save the results, including the closest match sequence and its alignment score.

----------------------------------------------------------

### Approach and Logic Details. Basically, below describe the steps taken for going from A --🡪 B

The following steps outline the process flow from reading the input files to generating the desired output. These steps are embedded into the code.

#### Step 1: Reading and Validating Input Files

Objective: Ensure that the `dog\_breeds.fa` (database) and `mystery.fa` (query) files are in the correct format and contain valid data.

Details:

- Read the sequences from both input files using `Bio.SeqIO`.

- Validate that the files are in FASTA format.

- Handle any file errors (e.g., file not found, empty file, invalid format).

- Validation:

- Check that the files exist.

- Ensure that the data in both files is correctly formatted (e.g., valid DNA sequences).

#### Step 2: Sequence Alignment (Using `pairwise2`)

Objective: Align the test sequence (`mystery.fa`) against the sequences in the database (`dog\_breeds.fa`).

Details:

- Use the `pairwise2` module to perform a global alignment of the test sequence with each sequence in the database.

- Store the alignment score and sequence match.

- The best match will be determined by the highest alignment score.

- Testing:

- Test the alignment function for different sequence pairs (test and database) to ensure correct alignment and scoring.

#### Step 3: Match Scoring

Objective: Calculate the similarity score for each alignment.

Details:

- For each alignment, extract the score and check if it is the best score seen so far.

- Track the sequence with the highest score, which will be the closest match.

- Testing:

- Verify the score calculation logic for known sequences.

- Test with a variety of sequence lengths and complexities.

#### Step 4: Output Results

Objective: Output the closest match sequence and the alignment score.

Details:

- Save the closest match's identifier and score to a text file (`classification\_results.txt`).

- Optionally, output the alignment details or the sequence comparison.

- Testing:

- Test that the output is written correctly.

- Verify that the output matches expected results.

#### Step 5: (Stretch Goal) Probabilities and p-values

- \*\*Objective\*\*: Calculate the probability of the test sequence matching each sequence in the database.

- \*\*Details\*\*:

- Implement a method to calculate probabilities of match based on alignment score.

- Compute p-values to determine the statistical significance of the alignment.

- \*\*Testing\*\*:

- Test the probability and p-value calculations with known values and sequences.

#### \*\*Step 6: (Stretch Goal) Phylogeny\*\*

- \*\*Objective\*\*: Construct a phylogenetic tree based on the sequence alignments.

- \*\*Details\*\*:

- Use tools like `scipy` or `Biopython` to generate a tree based on the alignment results.

- Visualize the tree using a plotting library (e.g., `matplotlib`).

- \*\*Testing\*\*:

- Test tree construction with known datasets.

- Validate the accuracy of tree visualization.

### \*\*4. Testing and Validation (Specific Tests)\*\*

Here are specific tests for each step of the code:

1. \*\*Input Validation Tests\*\*:

- Test with valid FASTA files containing multiple sequences.

- Test with an invalid file (e.g., incorrect format or missing file).

- Test with an empty file.

2. \*\*Alignment Function Tests\*\*:

- Test with known pairs of sequences and verify the alignment score.

- Test edge cases such as very short or very long sequences.

3. \*\*Match Scoring Tests\*\*:

- Test the match scoring with sequences that are identical, partially matching, or completely different.

- Ensure the correct sequence is returned with the highest score.

4. \*\*Output Tests\*\*:

- Verify the output file is created correctly.

- Check that the correct match and score are written to the output file.

5. \*\*Stretch Goal (Probability) Tests\*\*:

- Test the probability calculation with known match scenarios.

- Test edge cases with extreme probabilities (e.g., very low or very high).

6. \*\*Stretch Goal (Phylogeny) Tests\*\*:

- Test tree construction with a set of known sequences.

- Validate that the tree is correctly visualized and reflects expected relationships.

---

This should give you a good structure to proceed with coding and testing later. Let me know if you'd like further adjustments!