Pheno.AI

# Home Assignment - Student Data Engineer

## Overview

Pheno.AI is a data company targeting to personalized medicine based on large data sets collected worldwide. The company possesses the world's deepest phenotype datasets of healthy and diseased individuals and develops advanced AI algorithms to analyze them.
Our datasets range from DNA sequencing to blood tests, in different formats on different files, from small files to large files and images.
Our main challenge is how to take all this unstructured data from files/APIs, to structured normalized processed data.

## Task

You are tasked with developing an ETL (Extract, Transform, Load) system to process genetic data, specifically DNA sequences. The inputs are the exam results of a participant in a study of a DNA test. The output is one JSON file which is a combination of the process of two files, where the identifier of the JSON is the participant Id. The files are processed separately, but the output should be updated in a single merged JSON.
The system can process 2 types of files:

1. Text file (.txt), with the DNA sequence of the participant (example input on next section), with name convention of {participant_id}_dna.txt
2. JSON file (.json) with metadata of the test of the participant (example input on next section), with name convention of {participant_id}_dna.json

The program should identify the files and process each differently. How to process each file will be described below.
The input argument is a JSON, having two fields:

1. context_path: the path of input files. Should contain two files as described above.
2. result_path: the path where you should write the output JSON file. You should override the file if such already exists. After the processed is done- directory should have single JSON file.

Input:

```
{
    "context_path": "FILES_PATH",
    "results_path": "FILES_PATH/out/",
}
```

```
Example:
{
    "context_path": "./f3324a99-8a63-4ada-9d1d-562f84c7636d/",
    "results_path": "./f3324a99-8a63-4ada-9d1d-562f84c7636d/out/",
}
```

Notes On the File Input:

1. The fields in the example are required.
    a. The program should exit with an error code if the input is not in the format above.
2. context_path must:
    a. Be a valid path
    b. Files inside must have an extension of either ‘.txt’ or ‘.json’ only.
    c. The partial files name (without the folders and/or extension) is a valid uuid. This uuid is the participant_id
    d. In the very same folder, there must be files with the same uuid file name but extensions .txt, .json
3. `results_path` must:
    a. Be a valid path
    b. The file **may or may not exist**. If the file already exists, it should be overridden
4. The program should be able to handle data being sent twice per participant. The data should be overridden.

**The output is at the end of the task description.**

## JSON File Process Output

The JSON file contains metadata of the test and subject. The processing here is as follows:

1. Remove all related sensitive data related to the patient- specifically, every field which starts with “_”.
2. Validate all the values lengths are under 64 length
3. The participant must be at least 40 years old. All other dates should be between [2014-2024].

Example input:

```
{
  "test_metadata": {
    "test_id": "DNA123456",
    "test_type": "Genetic Analysis",
    "date_requested": "2024-12-01",
    "date_completed": "2024-12-10",
    "status": "Completed",
    "laboratory_info": {
      "name": "Genomics Lab Inc.",
      "certification": "CLIA Certified"
    }
  },
  "sample_metadata": {
    "sample_id": "SAMP987654",
    "sample_type": "Saliva",
    "collection_date": "2024-12-01",
    "data_file": <PATH_TO_TXT_FILE>,
  },
  "analysis_metadata": {
    "platform": "Illumina NovaSeq 6000",
    "methodology": "Whole Genome Sequencing",
    "coverage": "30x",
    "reference_genome": "GRCh38",
    "variants_detected": {
      "total": 5000,
      "pathogenic": 15,
      "likely_pathogenic": 8,
      "benign": 4977
    }
  },
  "individual_metadata": {
    "_individual_id": "IND123456",
    "_name": "John Smith",
    "date_of_birth": "1985-05-15",
    "gender": "Male",
    "ethnicity": "Caucasian",
    "family_history": {
      "diseases": [
        {
          "name": "Breast Cancer",
          "relation": "Mother",
          "age_at_diagnosis": 50
        },
        {
          "name": "Type 2 Diabetes",
          "relation": "Father",
          "age_at_diagnosis": 55
        }
      ]
    }
  }
}
```

The output should be in the same structure and file after the validation and reduction described above.

## TXT File Processing

*An explanation of DNA is in the appendix at the end.*
The text file contains DNA sequences. Each line describes a single sequence. You need to extract the following features:

1. For each sequence: GC content of a DNA sequence
    a. The GC content is the percentage of nucleotides in a DNA sequence that are either Guanine (G) or Cytosine (C).
2. For each sequence: Compute Codon Frequency for Each Sequence:
    a. Calculate the frequency of each codon (triplet of nucleotides) in each DNA sequence.
3. For the entire file- Compute the most frequent codon of all sequence.
4. For the entire file- Longest Common Subsequence (LCS) of all sequence combinations
    a. Find the longest common subsequence between any sequences in the file

*Example output on next page.*

<u>Notes On the .txt File Input:</u>

1. You can assume the input is valid
2. Each line should be processed separately.
3. Each line contains only A, G, C, T

### Text File Example (2 sequences to process):

```
ATCGATCGTAGCTAGCTAGCTGATCGATCGAT
ATCGGTAAATGCCTGAAAGATG
```

## TXT Process Output- Example

```
{
   "sequences": [
       {
           "gc_content": 46.88,
           "codons": {
               "ATC": 1,
               "GAT": 2,
               "CGT": 1,
               "AGC": 1,
               "TAG": 1,
               "CTA": 1,
               "GCT": 1,
               "CGA": 1,
               "TCG": 1
           }
       },
       {
           "gc_content": 40.91,
           "codons": {
               "ATC": 1,
               "GGT": 1,
               "AAA": 2,
               "TGC": 1,
               "CTG": 1,
               "GAT": 1
           }
       }
   ],
   "most_common_codon": "GAT",
   "lcs": {
       "value": "ATCG",
       "sequences": [
           1,
           2
       ],
       "length": 4
   }
}
```

## Final Output:

Notes:

- start_at and end_at are the last time of processing the data
- participant/_id is a random valid uuid based on the file name

```
{
    "metadata": {
        "start_at": "2024-05-01T13:37:39.348785+00:00",
        "end_at": "2024-05-01T13:37:40.086917+00:00",
        "context_path": "INPUT_PATH_HERE",
        "results_path": "OUTPUT_PATH_HERE",

    },
    "results": [
        {
            "participant": {
               _id: "f3324a99-8a63-4ada-9d1d-562f84c7636d"
             },
            "txt": {The json above under ‘TXT Process Output- Example’},
            "JSON": {The Json above under ‘JSON File Process Output’}
        }
    ]
}
```

# Appendix: Definitions and Explanations:

DNA sequence is a specific order of nucleotides that make up a strand of DNA (deoxyribonucleic acid). DNA is the genetic material found in all living organisms and many viruses. It contains the instructions needed for an organism to develop, survive, and reproduce.

The four types of nucleotides in DNA are:

a. Adenine (A)
b. Thymine (T)
c. Guanine (G)
d. Cytosine (C)

A DNA sequence is typically represented as a string of these letters. For example, short DNA sequence might look like this: ATCGGTAAATGCCTGA

Features:

1. GC content of a DNA sequence is the percentage of nucleotides in a DNA sequence that are either Guanine (G) or Cytosine (C). For example in ATCGGTAAATGCCTGA, the GC content is:
   100 x ( 4 + 3 ) / 16 = 43.75%
   Where:
   - 4 is the number of Guanine (G)in the DNA sequence,
   - 3 is the number of Cytosine (c) in the DNA sequence
   - 16 is the length of the sequence
2. Codon Frequency of a DNA sequence is explained by the following examples:
   The codon frequency of ATCGGTAAATGCCTGAAAGG is:
   {(ATC = 1), (GGT = 1), (AAA = 2), (TGC = 1), (CTG = 1)}
   Note: Always start with the first nucleotides and possibly ignore/remove the last remaining 1 or 2 nucleotides (in the above example GG are ignored/removed)
   Another example: the Codon frequency of
   ATCGATCGTAGCTAGCTAGCTGATCGATCGAT is:
   {(ATC = 1), (GAT = 2), (CGT = 1), (AGC = 1), (TAG = 1), (CTA = 1), (GCT = 1), (CGA = 1), (TCG = 1)}
3. LCS (Longest Common Subsequences)
   The longest common subsequence between two DNA sequences- a single longest subsequence that appears in two (or more) sequences. This is a classic problem in bioinformatics to find similarities between sequences.

For any questions please reply to the email (use Reply All).

Good luck!