Follow these instructions to use the Business Logic Layer - All code is written in python.

Front End instructions

1. You must have all the files saved in 1 directory – this should already be set up.
2. Import the module Main\_file.py in to your code
3. Use following functions as required

Functions To return required lists of records.

getAllGenes() – returns A-Z list of all genes

getAllProteins() – returns A-Z list of all proteins

getAllAccessions() – returns A-Z list of all accessions

getAllNCBIIDs() – returns A-Z list of all NCBI Identifiers

getAllChromosomeLocations() – returns ordered list of Chromosome Locations

1. Call this function to retrieve all associated record data.

getAllEntryData(keyword, type)

keyword – Any accession/gene/protein/NCBI ID/Chromosomal location

type - Front end must specify if entered record is Gene/protein/accession of NCBI ID.

This will return the following python variables

|  |  |  |
| --- | --- | --- |
| Variable name (literal) | Explanation of what variable contains | Variable type |
| parsedSequence  codingRegion  listOfIntrons  concatinatedIntrons  mrnaSequence  splitSequence  translatedAndAligned  codonFrequency  RestrictionEnzymes | Entire nucleic acid sequence  Coding region of nucleic acid sequence  List of Nucleic acid sequence strings of the introns  All introns stuck together in one string  Mrna sequence – (t’s replaced with u’s)  Mrna sequence split in to codons  Sequence translated in to AA and aligned with Nucleic acid sequence  Codon frequencies  Dictionary of associated RE and their respective sequences | String  String  List  String  String  List of codons  2Aligned Strings  Dictionary  Dictionary |

Example –

getAllEntryData(U49845,Accession)

Returns -

parsedSequence = “actgtgggggtcacgtcgta…”

codingRegion = “gtcacgtgta…”

listOfIntrons = “gtgcacgtgaaagtg”, “gtcgtgtgtgggta”, “gtcgtggggcaaaa”

concatinatedIntrons = “gtgcacgtgaaagtggtcgtgtgtgggtagtcgtggggcaaaa…”

mrnaSequence = “gugcacgugaaaguggucgugugugggugucguggggcaaaa….”

splitSequence = gug,cac,gug,aaa,gug,guc,gug,ugu,ggg,ugu,cgu,ggg,gca,aaa….”

translatedAndAligned = ‘gugcacgugaaaguggucgugugugggugucguggggcaaaa..’, ‘T--B--A--G--T--N--R..’

codonFrequencies = codons = {"uuu": 20% , "uuc": 30%, "uua": 20%, "uug": 30% …}

RestrictionEnzymes = ‘EcoRI’,’BamHI’,’BsuMI…’

1. Call this function to find cut sites

restrictionEnzymeCutSites(bases) –

bases – input the sequence you are looking for eg. tcgaa

This will return a dictionary of the format

(start cut point, end cut point : ‘within coding region/not within coding region’)

Example

restrictionEnzymeCutSites(‘tcgaa’)

Returns

{(127, 340 : “not in coding region”), (500, 1500: “in coding region”), (1600, 2100: “not in coding region”

Back end instructions

1. Import module main\_file.py
2. To calculate total codon frequency use function

Totalcodonfrequency() – This will return a dictionary containing the 64 codons and their respective frequencies. This can then be stored in the DB and call upon again by the middle layer.