Codon Optimization for the Development of DNA Vaccine

Submitted by

Aishwarya Saha (13000120033)

Ananya Saha (13000120035)

Arnab Chakraborty (13000120040)

Nibir Das (13000120052)

7th Semester
December, 2023

Submitted for the partial fulfillment for the degree of Bachelor of Technology in Computer Science and Engineering

Techno Main Salt Lake
EM 4/1, Salt Lake, Sector – V, Kolkata – 700 091.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our guide of the department of Computer Science and Engineering, whose role as project guide was invaluable for the project. We are extremely thankful for the keen interest he / she took in advising us, for the books and reference materials provided for the moral support extended to us.

Last but not the least we convey our gratitude to all the teachers for providing us the technical skill that will always remain as our asset and to all non-teaching staff for the gracious hospitality they offered us.

Place: Techno Main Salt Lake

Date: 06.12.23

Aishwarya Saha (13000120033)

Ananya Saha (13000120035)

Arnab Chakraborty (13000120040)

Nibir Das (13000120052)

Department of Computer Science and Engineering
Techno Main Salt Lake
Kolkata – 700 091
West Bengal, India.

APPROVAL

This is to certify that the project report entitled " Codon Optimization for the Development of DNA Vaccines" prepared under my supervision by *Aishwarya Saha (13000120033), Ananya Saha (13000120035), Arnab Chakraborty (13000120040) and Nibir Das (13000120052)* be accepted in partial fulfillment for the degree of Bachelor of Technology in Department of Computer Science and Engineering.

It is to be understood that by this approval, the undersigned does not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn thereof, but approves the report only for the purpose for which it has been submitted.

……. ……

Signature of
Internal Guide(s)

Signature
of the HOD

……..

Signature of External Examiner

Table of Contents

1 Introduction

1.1 Abstract

In the rapidly advancing landscape of scientific research, our project focuses on addressing the challenge of decoding complex DNA sequences. Within the field of Computer Science and Engineering (CSE), the specific issue at hand involves inefficient gene translation due to organisms' codon preferences influenced by tRNA levels, leading to slow processes, errors, and reduced protein yields. Our proposed solution employs Codon Optimization, aligning gene sequences with host preferences to enhance translation efficiency within the realm of CSE. Anticipated outcomes encompass faster processes, reduced errors, and increased protein yields, signifying a substantial stride in understanding genetic intricacies. With broad-reaching benefits, our project contributes to more effective bioinformatics applications and marks a pivotal advancement in scientific research.

1.2 Problem Domain

The business domain of our project is Biotechnology Research and the high level technical domain for the problem is Gene Regulatory Network Analysis.

1.3 Glossary

DNA	Hold information on how cell works
RNA	Act to transfer short pieces of information to different parts of cell, provide templates to synthesize into protein
Proteins	Form enzymes that send signals to other cells and regulate gene activity, form the body's major components (e.g. hair, skin, etc.)
Gene	Basic physical and functional units of heredity, specific sequences of DNA bases that encode instructions on how to make proteins.
Codon	Codons are hereditary units that contain / the coding information for one amino acid. Each codon consists of any three nucleotides (a triplet) of DNA symbols in ΣN.
Codon Optimization	The process of strategically modifying DNA sequences to enhance gene expression efficiency
DNA Vaccines	Vaccines that use genetic material from pathogens to stimulate an immune response

Motif	A short DNA sequence occurring as its part
Bioinformatics	The use of computational tools and techniques for the analysis and interpretation of biological data

2 Problem Definition

2.1 Scope

There are vast DNA sequences with different combinations that play vital functions. Different organisms have distinct codon preferences, accordingly, if we introduce any foreign gene into the organism, it may lead to low translational efficiency, errors, and reduced protein yield. Hence, our objective is to align the gene sequence with the codon preference thus enhancing the translational efficiency. The project aims to overcome these challenges by implementing Codon Optimization techniques.

2.2 Exclusions

Following are some of the exclusions that are beyond the scope of this project:

- Experimental vaccine testing: Actual testing of the DNA vaccines on live organisms or conducting experimental trials to assess the vaccine's effectiveness in vivo is outside the scope of this project.
- Secondary structure prediction: Prediction of secondary protein structures resulting from the codon optimization is excluded from the project scope.
- Population-specific codon preferences: Fine-tuning codon optimization based on population-specific preferences is not within the project scope.

2.3 Assumptions

Here, are some of the assumptions listed for our project :

- No unforeseen genetic factors: We are assuming here that there are no unidentified genetic factors or complexities that significantly impact codon usage beyond the current state of knowledge.
- Stable codon preferences over time: We are also assuming a level of stability in codon preferences over a relevant time frame since that evolutionary changes may impact preferences in the long term.

3 Related Studies

3.1 Basic Terminologies

DNA (Deoxyribonucleic Acid) :
Deoxyribonucleic acid (DNA) is a nucleic acid that contains the genetic instructions for the development and function of living organisms. All known cellular life and some viruses contain DNA. The main role of DNA in the cell is the long term storage of information. It is often compared to a blueprint, since it contains the instructions to construct other components of the cell, such as proteins and RNA molecules. The DNA segments that carry genetic information are called genes, but other DNA sequences have structural purposes, or are involved in regulating the expression of genetic information.

The DNA double helix is held together by hydrogen bonds between the bases attached to the two strands. The four bases found in DNA are adenine (A), cytosine (C), guanine (G) and thymine (T).

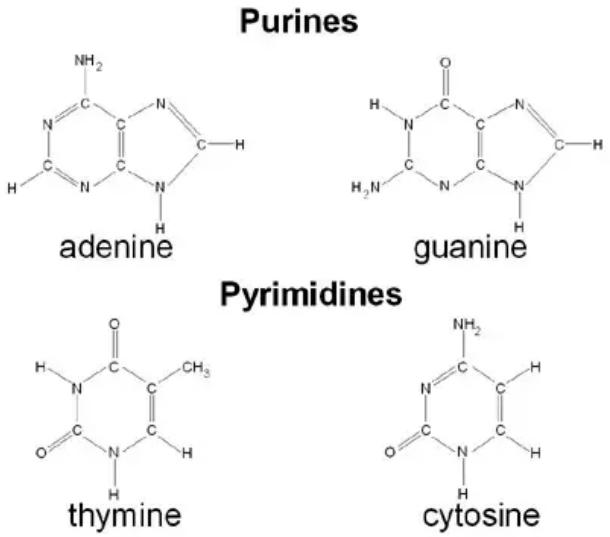

Fig 1: Four bases of DNA

Motif:
A motif is a short DNA sequence of length l, denoted as M = m1m2 m3……ml , mi ΣN ,1 ≤ i ≤ l A motif occurs as part of a long DNA sequence and as such will correspond to a sequence of codons

RNA (Ribonucleic Acid) :
Ribonucleic acid (RNA) is a nucleic acid polymer consisting of nucleotide monomers. RNA nucleotides contain ribose rings and uracil unlike deoxyribonucleic acid (DNA), which contains deoxyribose and thymine. It is transcribed (synthesized) from DNA by enzymes called RNA polymerases and further processed by other enzymes. RNA serves as the template for translation of genes into proteins, transferring amino acids to the ribosome to form proteins, and also translating the transcript into proteins.
RNA is primarily made up of four different bases: adenine, guanine, cytosine, and uracil.

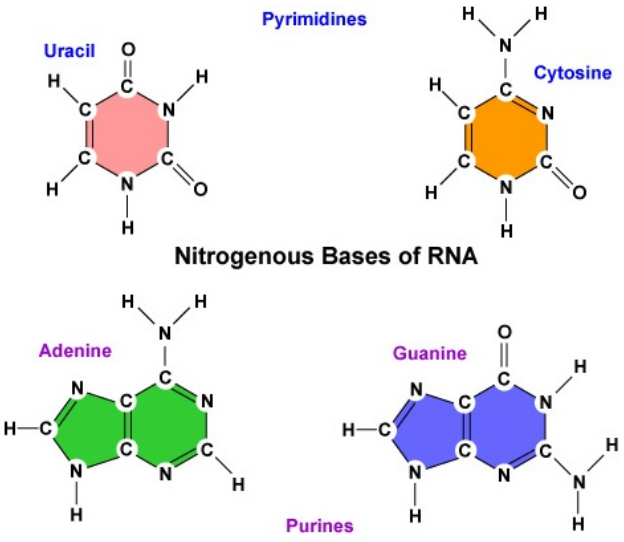

Fig 2: Four bases of RNA

Amino Acids :

Amino acids are the basic structural building units of proteins. They form short polymer chains called peptides or longer chains either called polypeptides or proteins. The process of such formation from an mRNA template is known as translation which is part of protein synthesis. Twenty amino acids are encoded by the standard genetic code and are called proteinogenic or standard amino acids.

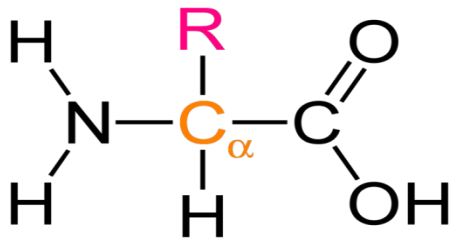

Fig 3: Structure of α-amino acid molecule

The general structure of an α-amino acid molecule, with the amino group on the left and the carboxyl group on the right. In the structure shown to the above, the R represents a side chain specific to each amino acid. The central carbon atom called Cα is a chiral central carbon atom to which the two termini and the R-group are attached. Amino acids are usually classified by the properties of the side chain into four groups. The side chain can make them behave like a weak acid, a weak base, a hydrophile, if they are polar, and hydrophobe if they are nonpolar.

The alphabet for the amino acid sequence is ΣA = {A,R,N,D,C,Q,E,G,H,I,L,K,M,F,P,S,T,W,Y,V,STOP}, each letter indicating one of the 20 amino acids, and STOP indicating the STOP codon.

Central Dogma :
The central dogma of molecular biology states that:
DNA acts as a template to replicate itself, DNA is also transcribed into RNA, and RNA is translated into protein [25].

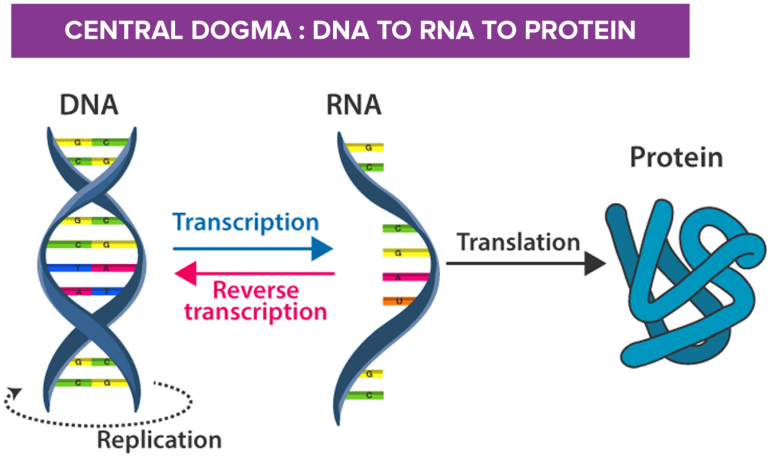

Fig 4: Central Dogma of Molecular Biology

Codon :
Codons are hereditary units that contain the coding information for one amino acid. Each codon consists of any three nucleotides (a triplet) of DNA symbols in ΣN. The information is first transcribed into messenger RNA,which has a sequence of bases complementary DNA from which it is copied.
DNA and mRNA have only four different bases (A,C,T,G in DNA and A,C,G,U in RNA)whereas proteins contain 20 amino acids.

When designing a DNA vaccine for a species, it is often desirable to have those codons that occur more frequently in that species. Therefore, for each amino acid we generally choose the most frequent codon, henceforth referred to as best codon.

Codon Usage Bias :
Codon usage bias is the preferential or non-random use of synonymous codons, a ubiquitous phenomenon observed in bacteria, plants and animals. Different species have consistent and characteristic codon biases. Codon bias varies not only with species, family or group within the kingdom, but also between the genes within an organism. Codon usage bias has evolved through mutation, natural selection, and genetic drift in various organisms. Codon-usage bias is a critical factor determining gene expression and cellular function by influencing diverse processes such as RNA processing, protein translation and protein folding. Codon usage bias reflects the origin, mutation patterns and evolution of the species or genes.

In general, codons can be grouped into 20 disjoint families, one family for each of the standard amino acids, with a 21st family for the translation termination signal. Each family in the universal genetic code contains between 1 and 6 codons. Where present, alternate codons are termed as synonymous.

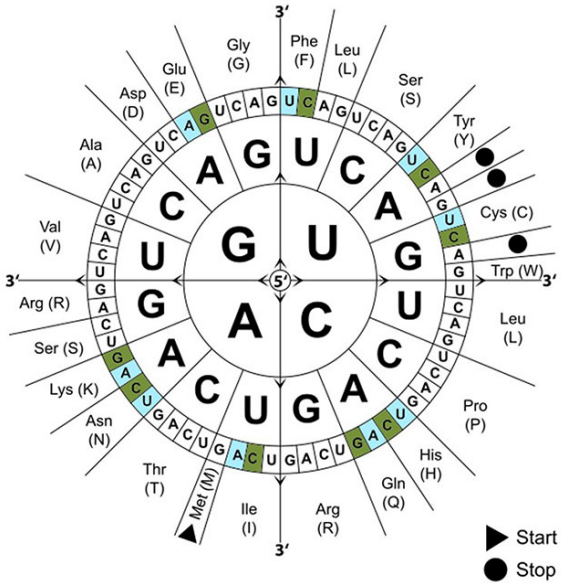

Fig 5 : Codon Usage/Bias

Codon Adaptation Index :
Codon Adaptation Index (CAI) is a simple, effective measure of synonymous codon usage bias.The index uses a reference set of highly expressed genes from a species to assess the relative merits of each codon, and a score for a gene is calculated from the frequency of use of all codons in that gene. The index assesses the extent to which selection has been effective in molding the pattern of codon usage. In that respect it is useful for predicting the level of expression of a gene, for assessing the adaptation of viral genes to their hosts, and for making comparisons of codon usage in different organisms. The index may also give an approximate indication of the likely success of heterologous gene expression. [11]

We looked at different methods used to improve how genes are read, focusing on codon optimization. After a comparative study, we figured out what each method does well and where they may fall short. This helped us create our own method, designed specifically to solve codon optimization more efficiently for developing DNA vaccines.

Codon Optimization :
Codon optimization is the process of modifying the DNA sequence of a gene to improve the expression of the corresponding protein in a particular host organism. This is done by replacing rare codons with more frequently used codons in the host organism. The goal of codon optimization is to increase the amount of protein that is produced by the host organism. [23]
Codon optimization is a technique that changes the coding sequence of a gene to increase

the expression of the protein it encodes. This is done by substituting rare codons for more frequent codons. This is important because the frequency of a codon can affect how easily it is translated by ribosomes.[24]
Codon optimization is a gene engineering approach to increase protein production. It is done by altering the codon sequence of a gene without changing the amino acid sequence. This is possible because most amino acids are encoded by more than one codon. Codon optimization is often used to improve the expression of recombinant proteins in bacteria or other cells.[25]

DNA Vaccines :
DNA vaccines are those vaccines that use genetic material from pathogens to stimulate an immune response in the host body against them. DNA vaccines have revolutionized the field of vaccine technology by demonstrating the ability to induce humoral and cellular immune responses in experimental animals and humans. Immunization of animals with plasmid DNA encoding a protein antigen was an accidental observation that eventually led to a novel strategy of immunization. DNA vaccines, also known as "naked DNA" or "nucleic acid" vaccines, encode the antigens of pathogenic organisms, including viruses, bacteria, fungi, and parasites. Although DNA vaccines have been successful in generating strong immune responses in smaller animal models, such as mouse, they have not been as efficient in larger species, such as primates and humans.

Need of Codon Optimization :
The expression of proteins from DNA introduced into heterologous host organisms is a common approach used for the generation of large quantities of protein for biological research. Bacteria,yeast,and baculovirus expression systems are examples of those commonly used to generate proteins of interest from introduced expression vectors or DNA introduced by genome integration.

A potential problem with such approaches is codon usage in different organisms, such that particular codons in the host gene sequence may not be matched with efficient translation of these codons in organisms used for heterologous expression. Optimal codon usage therefore refers to the need to modify DNA sequence in such a way that codons are selected to maximize the usage in the expression organism while maintaining the host gene's encoded protein sequence [12,13,14,15,16,17].
Another important issue where DNA sequence changes are necessary is in the use of DNA vaccines [18]. The effectiveness of such vaccines not only relies on the protein expressed but also on the ability of the DNA to elicit an effective immune response. In this respect, it has been demonstrated that certain short DNA motifs may be immunostimulatory or immuno-inhibitory and that modification of vector DNA sequence to optimize motif frequency can enhance immune response and consequently the effectiveness of the vaccine [19,20].

3.2 Algorithm Analysis

3.2.1 **Dynamic Programming** [3]

Description :

Dynamic Programming is a fundamental programming technique, applicable to great advantage where the input to a problem spawns an exponential search space in a structurally recursive fashion. If sub problems are shared and the principle of sub-problem optimality holds, DP can evaluate such a search space in polynomial time. Classical application examples of DP are optimal matrix chain multiplication or the triangulation of a convex polygon (Cormen et al., 1990).
For very good reason, Dynamic Programming is the most popular paradigm in computational molecular biology. Sequence data — nucleic acids and proteins — are determined on an industrial scale today. The desire to give a meaning to these molecular data gives rise to an ever increasing number of sequence analysis tasks, which naturally fall into the class of problems outlined above. Dynamic Programming is used for assembling DNA sequence data from the fragments that are delivered by automated sequencing machines (Anson and Myers, 1997), and to determine the exon-intron structure of eukaryotic genes (Gelfand and Roytberg, 1993). It is used to infer function of proteins by homology to other proteins with known function (Needleman and Wunsch, 1970; Smith and Waterman, 1981)[5], and it is used to predict the secondary structure of functional RNA genes or regulatory elements (Zuker, 1989).

Algorithm :

1. Initialization

   Upper border: position(i,0) represents the alignment of x1……..i to the empty prefix of y.
   That is, the prefix x1…….i has been matched with i gaps in y.
   With simple linear gap costs, the score is – d . i . The traceback pointer (i,0) points to (i1,0) .

```
for ( int i = 0; i <= n; i++ ) {
F[i][0] = - d * i ;
B[i][0] = new Traceback2 ( i-1, 0 ) ;
}
for ( int j = 1; j <= m; j++ ) {
F[0][j] = - d * j ;
B[i][0] = new Traceback2 ( 0, j-1 ) ;
}
```

2. Filling in the Matrix

   Position (i, j) may be reached
   • from (i-1, j-1) with a match, adding score[xi][yj] to the score;
   • from (i-1, j) with a gap in y, subtracting d from the score; or
   • from (i, j-1) with a gap in x, subtracting d from the score;

3. Traceback

The traceback B[i][j] points to the source of the maximal resulting score F[i][j]; Thus:

```
for ( int i = 0; i <= n; i++ )
for ( int j = 1; j <= m; j++ ) {
int s = score[seq1.charAt(i-1)][ seq2.charAt(j-1)];
int val = max ( F[i-1][j-1] + s, F[i-1][j] – d, F[i][j-1] - d ) ;
F[i][j] = val;
if ( val = = F[i-1][j-1] + s)
B[i][j] = new Traceback2 ( i-1, j-1 ) ;
else if ( val = = F[i-1][j] – d)
B[i][j] = new Traceback2 ( i-1, j ) ;
else if ( val = = F[i][j-1] – d )
B[i][j] = new Traceback2 ( i, j-1) ;
}
B0 = new Traceback2 ( n, m) ;
```

The start B0 of the traceback cell is (n, m).

Limitations :

- Dynamic programming can be computationally expensive, especially for long genes or large reference codon usage tables. This can lead to increased processing time and memory requirements.
- While dynamic programming can be adapted to different organisms and expression systems, it may require significant modifications and customizations for specific applications or objectives.

3.2.2 **Pattern Matching** [1]

Description :

In 2003, A Pattern Matching Algorithm for Codon Optimization and CpG Motif-Engineering in DNA Expression Vectors was proposed by Ravi Vijaya Satya, Amar Mukherjee, Udaykumar Ranga. In this system, a DNA sequence for a given amino acid sequence is synthesized, and graph theory is applied to maximize desirable motifs and minimize undesirable motifs. The approach uses an algorithm to locate a given set of immuno-modulatory motifs in the DNA expression vectors corresponding to a given amino acid sequence and maximize or minimize the number and context of the immuno-modulatory motifs in the DNA expression vectors.

MOTIF : A motif is a short DNA sequence of length l, denoted as M = m1m2m3……ml , mi $\in$ ΣN ,1 $\leq$ i $\leq$ l A motif occurs as part of a long DNA sequence and, as such, will correspond to a sequence of codons.

Input :

1. An amino acid sequence SA = a1a2a3a4…….aj
2. A list of motifs M = {M1, M2, M3, …….MK}
3. The motifs have a set of desirability values. Depending on the application desirability values of +1 and -1 are assigned to the motifs.

Algorithm :

I. Each motif from the given input list of motifs is taken to find out the amino acid search pattern corresponding to each of them.
II. We take each motif with high desirability value and perform the following steps:-
    A. First, we try to find out if the amino acid search pattern (AASP) obtained from the motif can be mapped to any of the amino acids present in the amino acid input sequence. If it is possible to map successfully, then we move on to the next step.
    B. The head and tail corresponding to the aasp are now checked against the amino acids that precede and succeed the amino acid (obtained from aasp mapping).
    C. If the head and tail could be successfully mapped to the preceding and succeeding amino acids, then it is referred to as a possible occurrence of the motif. The codons corresponding to an occurrence will be referred to as constituent codons for that occurrence.
III. For undesirable motifs, we consider all alternate codon combinations of each potential occurrence of an undesirable motif, and treat them as potential occurrences with a desirability value of zero.
IV. For amino acids that could not be mapped from desirable motifs, we replace the codons by the best codons of these amino acids.

Limitations:

- One of the major challenges associated with using pattern matching in bioinformatics is that, in most cases, the task isn't simply one of finding a match for a given pattern, but finding one or more matches quickly from large databases using affordable and readily available hardware.
- In addition, the task is often complicated by the need to identify patterns that are "similar" to a target pattern, but the concept of similarity aren't well defined from a programmatic and biological sense. The issues related to single pairwise sequence alignment, global versus local alignment, and multiple sequence alignment[2,4].

3.2.3 **One-to-One Codon Substitution** [26,27,28]

Description :

One-to-one codon substitution, also known as synonymous substitution, is a type of point mutation in which one nucleotide in a codon is replaced by another nucleotide, but the amino acid encoded by the codon remains the same. These mutations are often considered to be neutral, as they do not alter the protein

sequence. However, recent studies have shown that one-to-one codon substitutions can have subtle effects on protein structure and function.

Algorithm :

I. Input: The DNA sequence to be optimized.
II. Identify synonymous codons: Determine all the synonymous codons for each amino acid in the DNA sequence.
III. Calculate codon usage bias: Calculate the codon usage bias for the host organism. This is the frequency of each codon in the host organism's genome.
IV. Replace codons: Replace codons in the DNA sequence with synonymous codons that are more commonly used by the host organism. This is usually done by selecting the codon with the highest codon usage bias.
V. Output: The optimized DNA sequence.

Limitations:

- Limited applicability due to ignoring factors such as codon context and gene-specific constraints.
- Decreased mRNA stability : can decrease the stability of mRNA, which can lead to decreased protein expression.
- Affected protein folding : can affect protein folding, which can lead to misfolded proteins.
- Affected translation timing : can affect translation timing, which can lead to disruptions in cellular processes.
- Affected protein localization : can affect protein localization, which can lead to proteins being mislocalized to the wrong cellular compartment.

3.2.4 **Frequency-Based Codon Usage** [11,29,30,31]

Description :

Codon usage bias (CUB) is a phenomenon where certain codons are used more frequently than others in a given organism's genome. This bias can have a significant impact on gene expression, as codons that are used more frequently are typically translated more efficiently. Frequency-based codon usage bias algorithms are a type of codon optimization algorithm that aims to increase the expression of a gene by replacing codons with those that are more commonly used by the host organism.

Algorithm:

1. Initialize an empty dictionary codon_counts to store codon counts and another dictionary codon_frequencies to store codon frequencies.

2. Iterate through the DNA sequence in triplets of nucleotides, starting from the specified reading frame (if provided).
3. For each triplet:
    a. Extract the current codon.
    b. If the codon is not a stop codon:
        i. If the codon is not already present in codon_counts:
            1. Add the codon as a key to both dictionaries and initialize their values to 0.
        ii. Increment the values associated with the current codon in both dictionaries.
4. Calculate the total number of codons in the DNA sequence (excluding stop codons).
5. For each codon in codon_frequencies:
    a. Calculate the frequency of the codon by dividing its count in codon_counts by the total number of codons and multiplying by 100 to express it as a percentage.
    b. Update the dictionary entry for the codon in codon_frequencies with its frequency.
6. Return both codon_counts and codon_frequencies dictionaries.

Limitations :

There are certain limitations and disadvantages that should be considered:

- Complexity of biological phenomenon: Frequency-based analysis fails to capture the complex interplay of factors influencing codon usage bias.
- Context-specific: Frequency-based analysis often relies on a single reference dataset, neglecting context-specific codon usage preferences within organisms and tissues.
- Effective but may not consider the specific context of the target gene.

3.2.5 Codon Approximation Index (CAI) [11]

Description :

CAI is a widely used method for measuring the codon optimization of a gene for expression in a specific organism. It compares the relative abundance of synonymous codons in the gene with the reference set of highly expressed genes in the organism. It utilizes codon usage data and calculates a score reflecting similarity to a reference organism. Also, offers a balanced approach, considering both frequency and context.

Limitations :

- Deterministic nature may limit the algorithm's ability to explore a diverse range of sequences

- May not dynamically adapt to specific optimization goals during the sequence generation process.

3.2.6 Sequence Annealing [21]

Description :

Sequence Annealing modifies the sequence of a gene to increase its translation efficiency in a specific organism. By strategically replacing codons with those found more frequently in the target organism, the gene becomes more adapted for its host and can be expressed at higher levels. It involves iterative refinement of sequences through simulated annealing. Here we are considering temperature - higher temperatures increase the kinetic energy of the molecules, making it harder for them to bind together.

Limitation :

- Computational intensive and may not guarantee optimal solutions.

3.2.7 Genetic Algorithm [22]

Description :

Genetic Algorithms begin with a population of individuals, each representing a potential solution to the problem. These individuals are often encoded as chromosomes. Each individual is assigned a fitness score based on how well it meets the desired objective. This score serves as a measure of its suitability for survival and reproduction. Selected individuals undergo crossover, where genetic material is exchanged, creating offspring with a combination of their parents' characteristics. Random mutations are introduced into the offspring with a low probability, adding diversity and preventing the population from getting stuck in local optima. We repeat these steps until new generations are created with progressively better fitness scores.

Steps :

1. Initialization
2. Evaluation of fitness score
3. Select individuals having higher probability
4. Crossover is applied. It combines genetic material from two parents to create offspring with a mixture of their characteristics.
5. Repeat steps 2–5 until a desired convergence criterion is met.

Limitation :

- Powerful but may be computationally expensive and lack interpretability.

- Choosing the right parameters for the GA, like population size, selection pressure, mutation rate, etc., can significantly impact its performance.
- GAs can converge prematurely to suboptimal solutions, especially when population diversity is not maintained properly.

4 Project Planning

4.1 Software Life Cycle Model

In this project, we have embraced the **Iterative and Incremental Model** to capitalize on its flexibility and adaptability, which align seamlessly with the dynamic characteristics inherent in bioinformatics research. This model allows us to progressively develop and refine the Codon Optimization algorithm in response to evolving insights, stakeholder feedback, and emerging genetic data. Iterative cycles facilitate continuous improvement, ensuring the algorithm's efficacy in enhancing translation efficiency for DNA vaccine development within the Gene Regulatory Network Analysis domain. The incremental approach supports ongoing testing, validation, and integration of features, enabling a responsive and iterative development process that is crucial in the rapidly evolving field of computational biology.

Till now, we have completed 3 iterations for the following functionalities :

1. Default Optimization
2. Custom Optimization
3. User Feedback

After each iteration, functionalities are added one after the other in a cumulative manner. For each iteration, all the steps of planning, analysis, design, coding, and testing are being maintained.

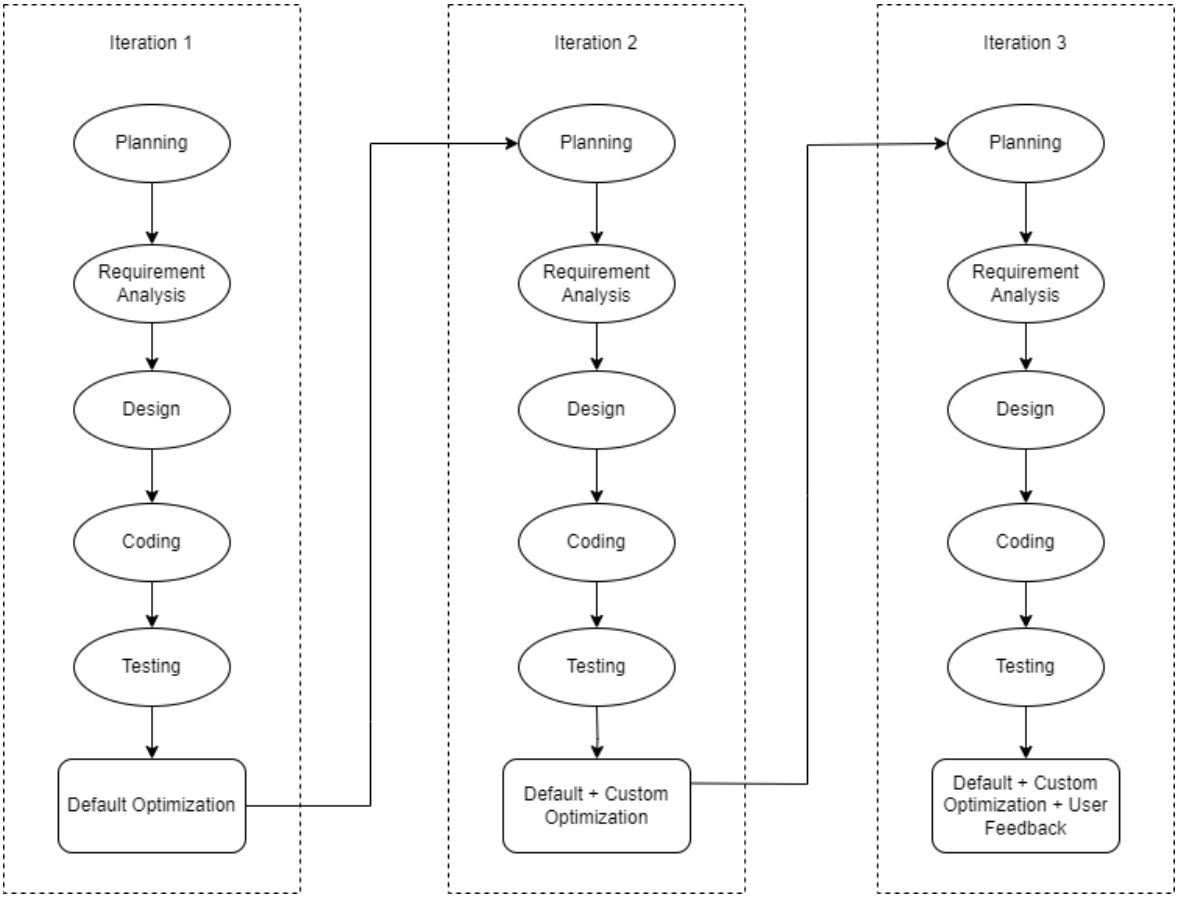

Fig 6 : Software Development Life Cycle of System

4.2 Scheduling

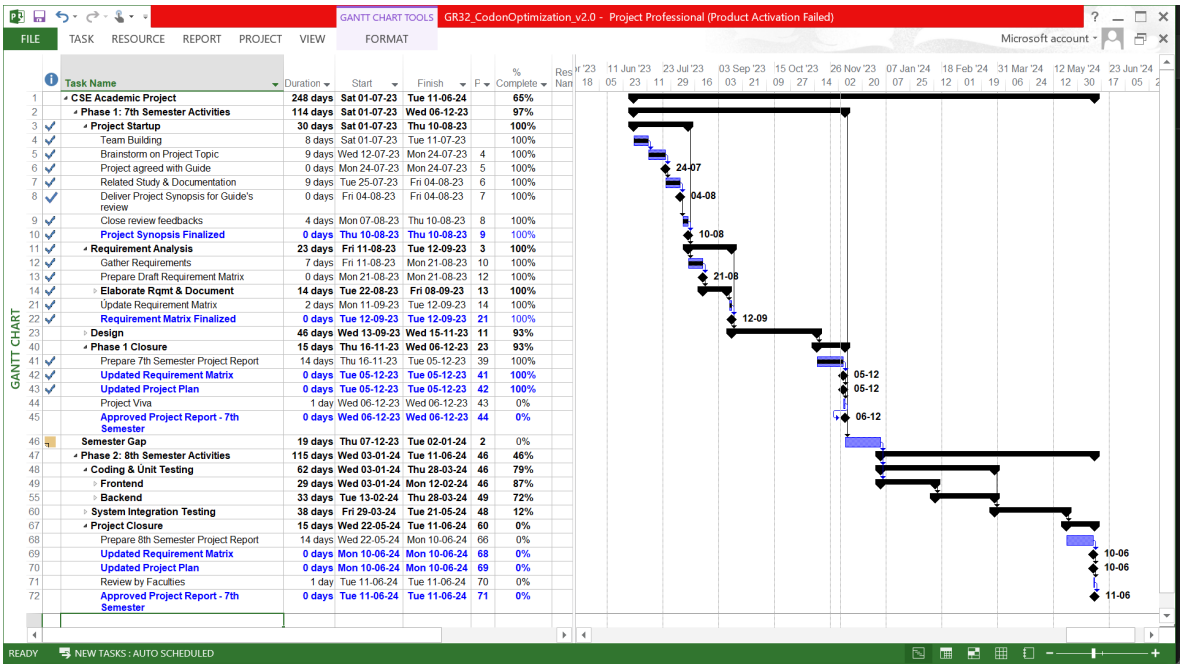

Fig 7 : Project Plan

4.3 Cost Analysis

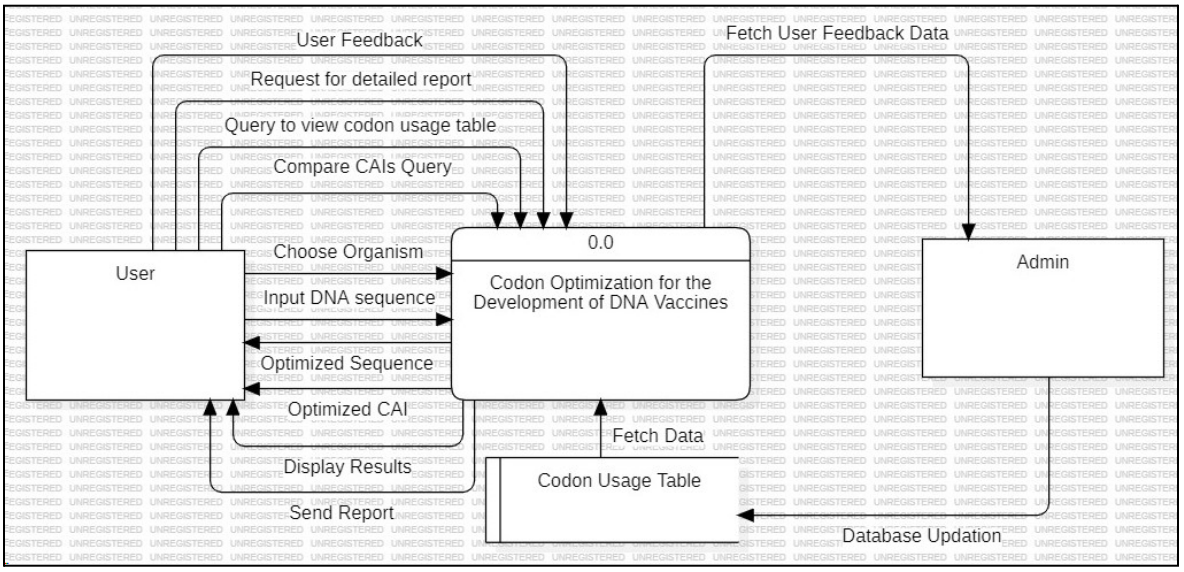

Fig 8 : DFD Level 0 of the System

Calculating total cost of the system considering the above designed DFD along with Function Point Metric and Constructive Cost Model, we have :
Total estimated cost for the above project with number of staff being 4 and lines of code

being 5051 (approx) is Rs. 332683.2944.

5 Requirement Analysis

5.1 Requirement Matrix

	Rqmt ID	Requirement Item	Requirement Analysis Status	Design Module	Design Reference (section# under project Report)	Test Case Number	Technical Platform of Implementation	Prototype prepared
6	USR-1.1	User is provided with two options.	Completed	USR	6.3.2.2	T-USR-1.1	Python	Yes
7	USR-1.2	User needs to select from any of the two options. (Custom , Default)	Completed	USR	6.3.2.2	T-USR-1.2	Python	Yes
8	USR-1.3	Upon selecting default optimize, corresponding page should open.	Completed	USR	6.3.2.3	T-USR-1.3	Python	Yes
9	USR-1.3.1	Allow user to choose organism from dropdown menu.	Completed	USR	6.3.2.3.1	T-USR-1.3.1	Python	Yes
10	USR-1.3.2	Allow user to enter DNA Sequence (Default)	Completed	USR	6.3.2.3.2	T-USR-1.3.2	Python	Yes
11	USR-1.3.3	User should click the optimize button (Default)	Completed	USR	6.3.2.3.3	T-USR-1.3.3	Python	Yes
12	PROC-1	Get choice of organism and the sequence from user	Completed	PROC	6.3.3.1	T-PROC-1	Python	Yes
13	PROC-1.1	Process codon optimization (Default)	Completed	PROC	6.3.3.2		Python	Yes
14	PERF-1	Should provide quick response during optimization	In-progress	PERF	6.3.5.1		Python	Yes
15	ERR-1	Display error message in case of inappropriate input	Completed	ERR	63.6.1	T-ERR-1	Python	Yes
16	OUT-1	Display the correct result in case of appropriate input	Completed	OUT	6.3.4.1	T-OUT-1	Python	Yes
17	PROC-2	Should display result within a reasonable time	In-progress	PROC	6.3.3.5		Python	Yes
18	USR-1.4	Upon selecting custom optimize, corresponding page should open.	Completed	USR	6.3.2.4	T-USR-1.4	Python	Yes
19	USR-1.4.1	Prompt user to upload codon usage table in .csv format	Completed	USR	6.3.2.4.1	T-USR-1.4.1	Python	Yes
20	SCR-1	Codon Usage table must be securely handled	Open	SCR	6.3.7			No
21	USR-1.4.2	Prompt user to enter target CAI value	Completed	USR	6.3.2.4.2	T-USR-1.4.2	Python	Yes
22	USR-1.4.3	Prompt user to enter default CAI range	Completed	USR	6.3.2.4.3	T-USR-1.4.3	Python	Yes
23	USR-1.4.4	Prompt user to enter DNA sequence (Custom)	Completed	USR	6.3.2.4.4	T-USR-1.4.4	Python	Yes
24	USR-1.4.5	User should click the optimize button (Custom)	Completed	USR	6.3.2.4.5	T-USR-1.4.5	Python	Yes
25	PROC-2	Get codon usage table , target cai , default cai range and the sequence from user	Completed	PROC	6.3.3.3	T-PROC-2	Python	Yes
26	PROC-2.1	Process codon optimization (Custom)	Completed	PROC	6.3.3.4		Python	Yes
27	ERR-2	Display error message in case of inappropriate file format , sequence	Completed	ERR	6.3.6.2	T-ERR-2	Python	Yes
28	OUT-2	Display the correct result in case of appropriate input (Custom)	In-progress	OUT	6.3.4.1	T-OUT-2	Python	Yes
29	USR-1.5	Allow user to give feedback	Completed	USR	6.3.2.5	T-USR-1.5	HTML/JS	Yes
30	PERF-3	Should handle growing number of users	In-progress	PERF	6.3.5.2		HTML	Yes
31	PERF-4	User Interface should give clear instructions and labels	Open	PERF	6.3.5.3			No
32	ERR-3	Error messages should be user friendly	In-progress	ERR	6.3.5.4		HTML/JS	Yes
33	PERF-6	System should be compatible with commonly used browsers and devices	Completed	PERF	6.3.5.4	T-PERF-1	HTML	Yes
34	DOC-1	Provide comprehensive documentation for users, including instructions on how to use the system, file format requirements, and troubleshooting tips.	In-progress	DOC	6.3.8		HTML/CSS	Yes

Fig 9 : Requirement Matrix of System

5.2 Requirement Elaboration

5.2.1 Users should be able to slide through the organism names smoothly.

5.2.1.1 Ensure a smooth and user-friendly interface for browsing and selecting organisms, possibly through a well-organized dropdown list or search functionality.

5.2.2 Users are provided with two options.

5.2.2.1 Present users with two distinct options for the optimization process - customization with specific parameters or a default, straightforward optimization process.

5.2.3 User needs to select from any of the two options. (Custom , Default)

5.2.3.1 Upon the user's selection of custom or default optimization, direct them to the respective pages where they can proceed with their chosen method.

5.2.4 Upon selecting default optimize, the corresponding page should open.

5.2.4.1 When users opt for default optimization, direct them to the relevant default optimization page.

5.2.5 Allow user to choose organism from the dropdown menu.

5.2.5.1 Offer a dropdown menu for users to select the organism they want to optimize.

5.2.6 Allow user to enter DNA Sequence (Default)

5.2.6.1 Provide a field for users to input the DNA sequence they wish to optimize using default settings.

5.2.7 User should click the optimize button (Default)

5.2.7.1 Initiate the optimization process after the user clicks the 'Optimize' button.

5.2.8 Get choice of organism and the sequence from user

5.2.8.1 Retrieve and process the selected organism and sequence input from the user for default optimization.

5.2.9 Process codon optimization (Default)

5.2.9.1 Execute the codon optimization algorithm based on default parameters.

5.2.10 Should provide quick response during optimization

5.2.10.1 Ensure the system responds promptly during the optimization process, minimizing waiting times.

5.2.11 Display error message in case of inappropriate input

5.2.11.1 Present clear and informative error messages if the user input is incorrect or inappropriate.

5.2.12 Display the correct result in case of appropriate input

5.2.12.1 Show the optimized result promptly if the input data is correct.

5.2.13 Should display result within a reasonable time

5.2.13.1 Optimize system performance to ensure timely display of results after the optimization process.

5.2.14 Upon selecting custom optimize, the corresponding page should open.

5.2.14.1 Provide users with a specific page for custom optimization upon their selection.

5.2.15 Prompt user to upload codon usage table in .csv format

5.2.15.1 Enable users to upload a codon usage table in a secure .csv format to guide customized optimization.

5.2.16 Codon Usage table must be securely handled

5.2.16.1 Implement robust security measures to safeguard uploaded codon usage tables from unauthorized access.

5.2.17 Prompt user to enter target CAI value

5.2.17.1 Ask users to input their desired target Codon Adaptation Index (CAI) for customization.

5.2.18 Prompt user to enter default CAI range

5.2.18.1 Allow users to set a default range for the CAI, defining upper and lower limits for optimization criteria.

5.2.19 Prompt user to enter DNA sequence (Custom)

5.2.19.1 Provide a field for users to input the DNA sequence for customized codon optimization.

5.2.20 User should click the optimize button (Custom)

5.2.20.1 Initiate the customized codon optimization process upon user input.

5.2.21 Get codon usage table , target cai , default cai range and the sequence from user

5.2.21.1 Retrieve and process all necessary user inputs for the customized optimization process.

5.2.22 Process codon optimization (Custom)

5.2.22.1 Execute the codon optimization algorithm based on custom parameters.

5.2.23 Display error message in case of inappropriate file format , sequence

5.2.23.1 Present clear error messages if the uploaded file format or sequence is incorrect.

5.2.24 Display the correct result in case of appropriate input (Custom)

5.2.24.1 Show the optimized result promptly based on the customized parameters provided by the user.

5.2.25 Allow user to give feedback

5.2.25.1 Create a mechanism for users to provide feedback on their experience with the optimization process.

5.2.26 Should handle growing number of users

5.2.26.1 Design the system to efficiently accommodate an increasing number of users without performance degradation.

5.2.27 User Interface should give clear instructions and labels

5.2.27.1 Ensure the interface provides clear guidance for users through proper instructions and labeling.

5.2.28 Error messages should be user friendly

5.2.28.1 Design error messages that are easy to understand, guiding users on how to rectify mistakes or provide correct inputs.

5.2.29 System should be compatible with commonly used browsers and devices

5.2.29.1 Ensure the system functions seamlessly across various browsers (e.g., Chrome, Firefox) and devices (desktops, tablets, mobiles).

5.2.30 Provide comprehensive documentation for users, including instructions on how to use the system, file format requirements, and troubleshooting tips.

5.2.30.1 Supply detailed documentation guiding users on system usage, file format requirements, and troubleshooting tips.

6 Design

6.1 Technical Environment

6.1.1 Operating System

6.1.1.1 Windows

6.1.2 IDE:

6.1.2.1 VS Code

6.1.2.2 JupyterLab

6.1.3 Programming Language:

6.1.3.1 Python.

6.1.4 Codon Database

6.1.4.1 NCBI GenBank

6.1.4.2 NCBI Codon Usage Database

6.1.5 Data Visualization Tools:

6.1.5.1 Flask

6.1.5.2 Html

6.1.5.3 Css

6.1.6 Sharing and Collaboration:

6.1.6.1 Git

6.1.6.2 Github

6.1.7 Documentation:

6.1.7.1 Google Docs

6.1.7.2 Google Sheet

6.2 Hierarchy of Modules

6.2.1 User Module: (USR)

The user module mainly consists of the user interactive components from choosing the method of optimization to the end of getting a proper result.

6.2.2 Processing Module: (PROC)

A core algorithm is developed for codon optimization. Considered factors for optimizing the sequence are sequence ,the CAI range for the organism, Average target CAI for the organism, host organism.

6.2.3 Output Module: (OUT)

This contains the visualization of the optimized sequence and the optimized Cai Value with a proper user interface.

6.2.4 Performance Module: (PERF)

This is the module where the other modules are integrated to make it work as a proper system which will be much more scalable for more performance .

6.2.5 Documentation and Reporting Module: (DOC)

A detailed report on the optimization process, including input sequences, optimization parameters, databases and results are briefly mentioned.

6.2.6 Security Module: (SCR)

The data provided from the user must be secure and maintained within the database.

6.2.7 Error Module: (ERR)

The error module consists of the components of Error Handling and Validation

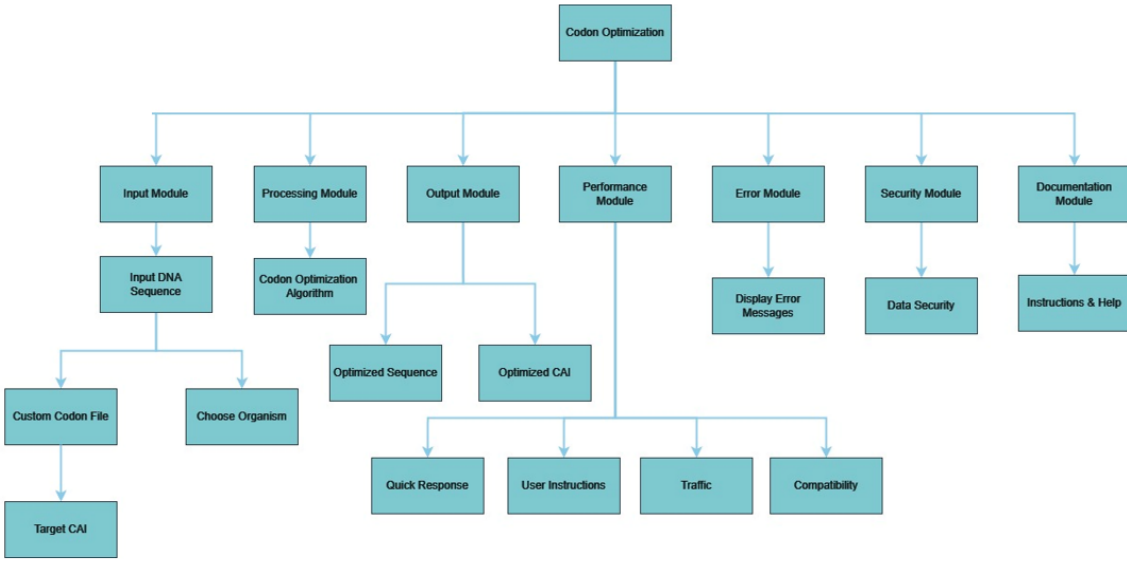

Fig 10: Hierarchical Diagram

6.3 Detailed Design

6.3.1 User Module

The user module is the part of the system where the main user interaction takes place it mainly consists of few parts:

6.3.1.1 Sliding through the default organism

The home page consists of a sliding window where the presented organism in the system is deployed. The user may check for the availability of the required organism in the window.

6.3.1.2 Choose optimization method

The user is given two options : Default optimization and Custom Optimization. Among these the user may proceed with their desired choice.

6.3.1.3 User chooses default optimization

The user chooses the option **Default Optimization** and then user is redirected to another page where the user has to input the following :

6.3.1.3.1 Choose organism

Here the user chooses the desired organism from the dropdown menu.

6.3.1.3.2 Enter DNA Sequence

The user enters the DNA sequence in this section that is to be optimized.

6.3.1.3.3 Click the optimize button

This is the submit button that ensures that the user inputs get submitted to the system.

6.3.1.4 User chooses custom optimization

The user chooses the option **Custom Optimization** and then user is redirected to another page where the user has to input the following :

6.3.1.4.1 Upload codon usage table

The user needs to upload the codon usage table for the target organism of the user. **The file needs to be in ".csv" format.**

6.3.1.4.2 Enter target CAI value

The user enters the average CAI value of the target organism in this section.

6.3.1.4.3 Enter default CAI range

The default CAI range is entered in this section. This is the range that optimized CAI value must reside.

6.3.1.4.4 Enter DNA sequence

The DNA sequence that is to be optimized is entered in this section.

6.3.1.4.5 Click the optimize button

This is the submit button that ensures that the user inputs get submitted to the system.

6.3.1.5 User feedback

The user can contact or give feedback to the admin by using the feedback section.

6.3.2 Processing Module

The processing Module has the following parts:

6.3.2.1 Fetch organism and Sequence

The system fetches the information that is provided by the user and feeds it into the optimization algorithm.

6.3.2.2 Process Codon Optimization for Default Optimization

The optimization algorithm is highlighted in this section. Here the previously fetched information is fed in the main algorithm.

6.3.2.3 Fetch Custom Table , Target CAI , Default CAI range and Sequence

The system fetches the information that is provided by the user while using the custom optimization and feeds it into the optimization algorithm. This mainly focuses on the added custom .csv file that is fed in the algorithm.

6.3.2.4 Process Codon Optimization for Custom Optimization

The optimization algorithm is highlighted in this section. Here the previously fetched information is fed in the main algorithm.The main standout here is that the custom .csv file is being transferred into the dictionary of the codon usage table.

6.3.2.5 Reasonable Time Complexity

The time complexity of the system must be reasonable such that the processing is carried out smoothingly.

6.3.3 Output Module:

The output module of the system contains two parts:

6.3.3.1 Display the result

The final value of the optimized sequence that isd obtained in the processing module that is being displayed in this module

6.3.4 Performance Module:

The performance Module has the following parts:

6.3.4.1 Quick response during Optimization

The overall optimization selection and input feature must be quick and smoothly done.

6.3.4.2 Handle growing traffic

The system must be able to handle the growing traffic with increasing users.

6.3.4.3 Clear instructions and Labels / User Guide

The system must have labels so that new users find it easy to navigate through the system.

6.3.4.4 Compatibility

The system must be compatible with every device irrespective of its operating system and its browser.

6.3.5 Error Module

The error module consists of the following parts:

6.3.5.1 Error message in case of inappropriate input

In the case where the user has given an inappropriate sequence the system checks it before feeding into the algorithm and it results in displaying an error message.

6.3.5.2 Error message in case of inappropriate file format , sequence

In the case where the user has given an incorrect format while using custom optimization that is without a .csv file then an error message is displayed as error in uploading the data.

6.3.6 Security Module

The security section is where a database is maintained in the system where the uploaded data is stored associated with the corresponding user credentials and all this has to be kept secure.

6.3.7 Documentation Module

The documentation is done in google docs with regular reports and system performance analysis.

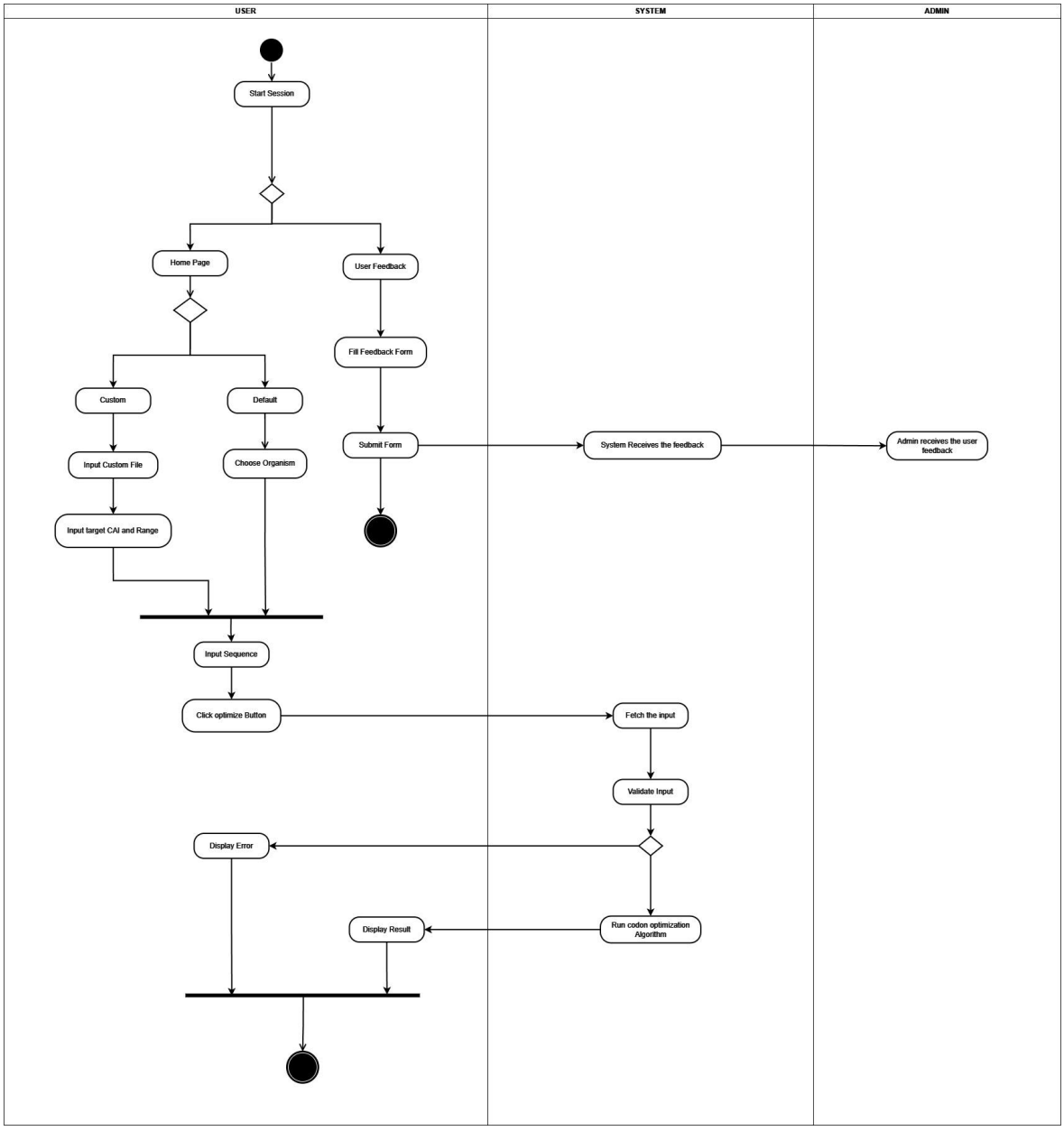

Fig 11: Activity Diagram

6.3.8 Comparative Analysis

Reasons why proposed algorithm is better than above studied algorithms:

- Proposed approach combines simple and efficient techniques, making it easier to implement and computationally less expensive compared to more complex algorithms like genetic algorithms.
- CAI calculation in this approach specifically targets the most important factor
-
- affecting translation efficiency, which is the codon usage bias of the target organism.
- Random mutation introduces diversity to explore different solutions, while two-point crossover helps refine existing solutions and combine favorable characteristics.
- This approach balances the need for maximizing protein expression with the need to maintain the original sequence and avoid unintended consequences.
- This approach is readily applicable to various organisms and expression systems, making it a versatile tool.

Following are the limitations of other algorithms :

- One-to-One - too simplistic and does not consider the context-dependent effects of neighboring codons
- Frequency-Based - focuses solely on individual codon frequencies and neglects other factors affecting translation
- Sequence Annealing - computationally expensive and may not be suitable for longer genes
- Genetic Algorithms - computationally costly and require careful parameter tuning, making them less user-friendly

	ALOGITHM							
ORGANISM		One to One	Frequency	CAI	Genetic	Sequence	Custom	Target CAI Range
	Human	1.00	5.61	0.89	5.64	1.35	0.80	0.8 - 1.0
	Wolf	1.00	3.70	0.86	4.59	1.76	0.40	0.4 - 0.6
	Cow	1.00	3.96	0.90	4.65	1.50	0.60	0.6 - 0.8
	Mushroom	1.00	3.70	0.93	4.21	1.35	0.80	0.8 - 1.0
	Gorilla	1.00	4.05	0.91	3.71	1.33	0.70	0.7 - 0.9

Fig 12: CAI Range table for various organisms

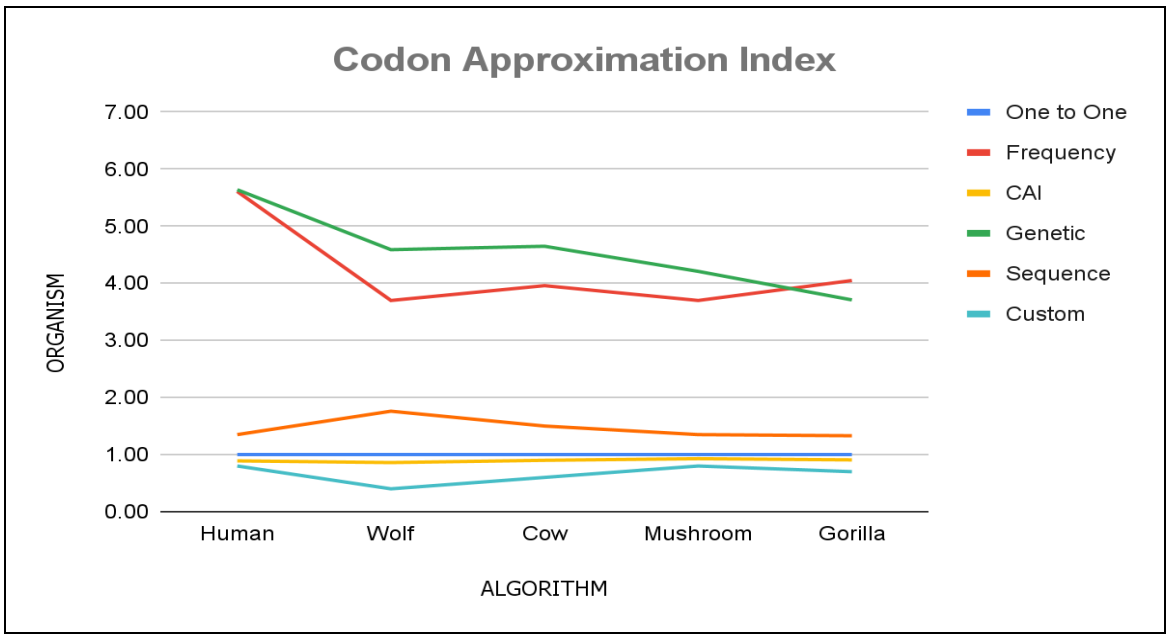

Fig 13 : Graphical Representation of Comparative Study with Example

6.3.9 Flowchart

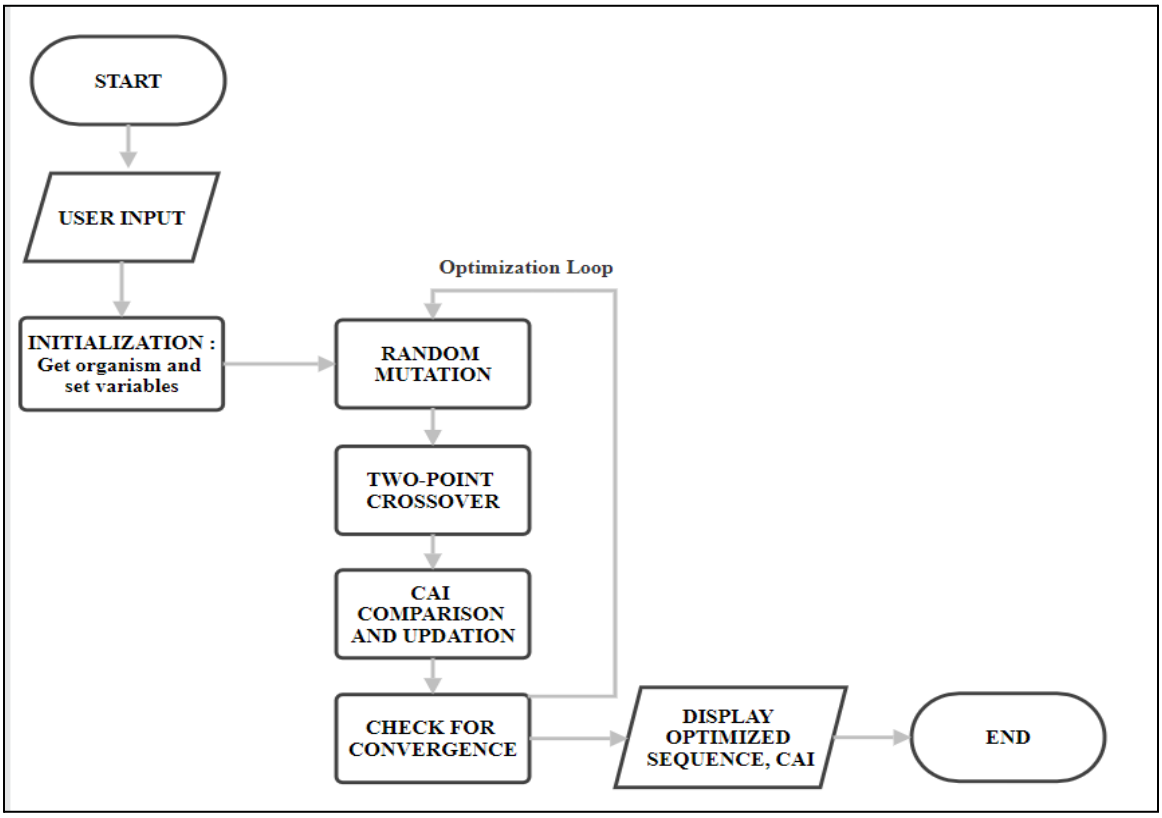

Fig 14 : Flowchart of Proposed Algorithm

6.3.10 Detailed Algorithm

6.3.10.1 Initialization:

6.3.10.1.1 Get organism choice and target sequence from the user.

6.3.10.1.2 Check if the input sequence is valid DNA.

6.3.10.1.3 Retrieve organism information from the dictionary (codon usage, target CAI, and default CAI range).

6.3.10.1.4 Initialize variables:

6.3.10.1.5 Set current sequence to the input sequence.

6.3.10.1.6 Set the best sequence and best CAI to the current sequence and its CAI.

6.3.10.2 Optimization Loop (Iterative Process):

Repeat for a maximum number of iterations:

6.3.10.2.1 Random Mutation:

- Apply random mutation to the best sequence.
- Calculate CAI for the mutated sequence.

6.3.10.2.2 Two-Point Crossover:

- Apply enhanced two-point crossover between the best sequence and the mutated sequence.
- Calculate CAI for the two crossovered sequences.

6.3.10.2.3 CAI Comparison and Updation:

- Update the best sequence and best CAI:
- If the mutated CAI is lower than the best CAI, update the best sequence and best CAI.
- If crossovered CAI1 is lower than the best CAI, update the best sequence and best CAI.
- If crossovered CAI2 is lower than the best CAI, update the best sequence and best CAI.

6.3.10.2.4 Check for Convergence:

- If the absolute difference between the best CAI and the target CAI is less than 0.001, break the loop.

7 Test Plan

Test-ID	Test Case Number	Description	Test Step	Expected Result	Status

T-USR-1 -	T-USR-1	To check smooth slide functionality through organism names.	Slide through the organism names	Sliding is done smoothly	Pass
T-USR-1 .1-	T-USR-1.1	To check if the user has two options for optimization.	Click on homepage	Two options are present	Pass
T-USR-1 .2-	T-USR-1.2	To check the selectivity of the options.	Select one of the options (Custom, Default)	Any one of the options is selectable at a time.	Pass
T-USR-1 .3-	T-USR-1.3	To check if on selecting Default Optimize, the corresponding page opens.	Clicking on Default Optimize option	The required page opens	Pass
T-USR-1 .3.1-	T-USR-1.3.1	To check if the desired organism can be selected from the dropdown menu.	Selecting an organism from the dropdown menu	The desired organism is selected	Pass
T-USR-1 .3.2-	T-USR-1.3.2	To check if a DNA sequence can be entered.	Entering the DNA sequence	The desired sequence is entered	Pass
T-USR-1 .3.3-	T-USR-1.3.3	To check if the optimize button is clickable.	Clicking on the optimize button	Optimize button is clicked	Pass
T-PROC-1-	T-PROC-1	To check if the user input is accepted.	Entering the DNA sequence and selecting the organism	User input is accepted	Pass

T-ERR-1 -	T-ERR-1	Input verification	Enter “ABCDEF” or “12345”	Error message: Not a Proper Sequence	Pass
T-OUT-1 -	T-OUT-1	To check if the correct output is shown for the Default Optimize page.	Entered “UGC CAC ACU AUG”	Result shown: UAGUUAG UAGUAG with optimized CAI: 0.8	Pass
T-USR-1 .4-	T-USR-1.4	To check if, on selecting Custom Optimize, a corresponding page opens.	Clicking on Custom Optimize option	The required page opens	Pass
T-USR-1 .4.1-	T-USR-1.4.1	To check if the user can upload a codon usage table in .csv format.	Clicking on Choose File button	The window opens for .csv file selection	Pass
T-USR-1 .4.2-	T-USR-1.4.2	To check if the user can enter the target CAI value.	Target CAI value is put: 0.8	The user is asked to enter a target CAI value. Upon receiving user input, it is accepted.	Pass
T-USR-1 .4.3-	T-USR-1.4.3	To check if the user can enter the default CAI range.	Default CAI range is put: 0.7-1.0	The user is asked to input the default CAI range. Upon receiving user input, it is accepted.	Pass
T-USR-1 .4.4-	T-USR-1.4.4	To check if a DNA sequence can be	Entering the DNA	The desired sequence is	Pass

		entered in the Custom Optimize page.	sequence in Custom Optimize page	entered	
T-USR-1 .4.5-	T-USR-1.4.5	To check if the optimize button is clickable in the Custom Optimize page.	Clicking on the optimize button in Custom Optimize page	Optimize button is clicked	Pass
T-PROC-2-	T-PROC -2	To check if the user input is accepted in the Custom Optimize page.	Entering target CAI, default CAI range, Dna sequence and codon usage table	User input is accepted	Pass
T-ERR-2 -	T-ERR-2	Input verification in Custom Optimize page.	Selecting codon.pdf file in Choose File option	Error message: Error uploading data	Pass
T-OUT-2 -	T-OUT-2	To check if the correct output is shown for the Custom Optimize page.	Entering all the inputs and selecting optimize button	Correct output is displayed	Suspended
T-USR-1 .5-	T-USR-1.5	To check if the user can give feedback.	Clicking on the feedback button.	Feedback page opens which is functional.	Pass
T-PERF-1-	T-PERF -1	To check if this system is compatible with commonly used browsers and devices.	System is opened through phone, laptop using google chrome and	The system works in those browsers and devices.	Pass

			firefox.		

8 Conclusion

8.1 Project Benefits

Some of our Project benefits are as follows:

- Host Organism Preference: Different organisms have different preferences for codon usage due to variations in their tRNA abundance and availability. Our project deals with efficiently recognized codon usage of the donor and translates the host organism's cellular machinery.
- Anticipating Genetic Modification: If the gene is being used for genetic modification or engineering purposes, codon optimization can make the gene more compatible with cloning techniques, gene synthesis, and other molecular biology methods.
- Translation Efficiency: By optimizing the codon usage to match the host's preferences, the translation process can be enhanced, leading to higher protein production levels.
- Protein Folding and Function: The speed and accuracy of translation can influence protein folding and functionality. Codon optimization improves the translation kinetics and fidelity.

8.2 Future Scope for improvements

Some of our future scope for improvement are as follows:

- We will research more efficient algorithms. This might involve exploring new machine learning techniques or optimizing existing ones.
- We would want to develop our project more, such that it will be able to handle large datasets at a time, and provide us with their respective solutions simultaneously. Integrating your codon usage analysis code with cloud systems is being considered.
- Connecting codon usage analysis with broader systems biology approaches, considering the interplay of genes, proteins, and other cellular components is also one of our future scope. This can provide a more holistic understanding of biological processes.

9 References / Bibliography

BOOKS:

- T. K. Attwood, and D. J. Parry-Smith, Introduction to Bioinformatics.
- Bryan Bergeron, The complete, practical guide to bioinformatics for life scientist Bioinformatics Computing.
- Holland, J. H. (1992). Adaptation in natural and artificial systems. MIT Press.
- Mitchell, M. (1998). An introduction to genetic algorithms. MIT Press.

- Goldberg, D. E. (2006). Genetic algorithms in search, optimization and machine learning. Addison-Wesley Professional.
- Haupt, R. L., & Haupt, S. E. (2004). Practical genetic algorithms. John Wiley & Sons.

RESEARCH PAPERS:

[1] Ravi Vijaya Satya, Amar Mukherjee and Udaykumar Ranga, "A pattern matching algorithm for codon optimization and CpG motif-engineering in DNA expression vectors," Computational Systems Bioinformatics. CSB2003. Proceedings of the 2003 IEEE Bioinformatics Conference. CSB2003, Stanford, CA, USA, 2003, pp. 294-305, doi: 10.1109/CSB.2003.1227330.

[2] Liu, Derong & Xiong, Xiaoxu & Dasgupta, Bhaskar & Zhang, H.. (2006). Motif Discoveries in Unaligned Molecular Sequences Using Self-Organizing Neural Networks. IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council. 17. 919-28. 10.1109/TNN.2006.875987.

[3] Taun D. Pham, James O'Connell, and Denis I. Crane, Constrained Codon Optimization by Dynamic programming, International Symposium, Hong Kong, October, 2004, p 20-22.

[4] K. Charter, J. Schaeffer, D. Szafron. Sequence Alignment using FastLSA. International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences. 2000

[5] Needleman, S.B. and Wunsch, C.D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology

[6] Sharp PM, Li WH. An evolutionary perspective on synonymous codon usage in unicellular organisms. J Mol Evol 1986;24(1-2):28-38

[7] Computational methods in molecular biology by Steven L. Salzberg, David B. Searls, Simon Kasif

[8] Genetic Algorithms By David E. Goldberg

[9] Codon usage bias by Sujatha Thankeswaran Parvathy, Varatharajalu Udayasuriyan, Vijaipal Bhadana

[10] Sharp PM, Li WH. The Codon Adaptation Index – a measure of directional synonymous codon usage bias, and its potential applications. Nucleic Acids Res 1987 Feb 11 ;15(3):1281-95

[11] C. H. Kim, Y. Oh, and T. H. Lee, Codon Optimization for High level

expression of human erythropoietin (EPO) in mammalian cells, Gene, 199(1997) 293-301.

[12] L. Wu L, M. A. Barry, Fussion protein vectors to increase protein production and evaluate the immunogenicity of genetic vaccines, Mol. Ther. 2(2000), pp. 288-297.

[13] M. L. Valencik, and J. A. Macdonald, Codon optimization markedly improves doxycycline regulated gene expression in mouse heart, Transgenic Res.,10(2001), pp 269-275.

[14] M. Fuller, and D. S. A Anson, Helper Plasmids for production of HIV-1 derived vectors, Hum, Gene, Ther., 12 (2001) ,pp. 2801-2093.

[15] A. Cid-Arregui, V. Juarez, and H. zur Hansen, A synthetic E7 gene of human papillomavirus type 16 that yields enhanced expression of the protein in mammalian cells and is useful for DNA immunization studies, J. Virol, 77 (2003), pp 4928-4937.

[16] E. M. Slimco, and H. A. Lester, Codon optimization of Caenorhabditis elegans GluCI ion channel genes for mammalian cells dramatically improves expression levels, J. Neirosci. Meth., 124 (2003), pp 75-81.

[17] S. Gurunathan, D. M. Klinman, and R. A. Seder, DNA vaccines: immunology, application, and optimization, Annu, Rev. Immunol., 18 (2000) 927-924.

[18] A. M. Krieg, CpG motifs in bacterial DNA and their immune effects, Annu. Rev. Immunol., 20 (2002) 709-760.

[19] A. M. Krieg, A. K. Yi, J. Schoor, and H. L. Davis, The role of CpG dinucleotides in DNA vaccine, Trends Microbiol., 6 (1998) 23-27.

[20] Computational methods in molecular biology by Steven L. Salzberg, David B. Searls, Simon Kasif

[21] Codon optimization with deep learning to enhance protein expression Hongguang Fu, Yanbing Liang, Xiuqin Zhong,corresponding author ZhiLing Pan, Lei Huang, HaiLin Zhang, Yang Xu, Wei Zhou, and Zhong Liu

[22] Developing a codon optimization method for improved expression of recombinant proteins in actinobacteria by Yutaka Saito, Naoto Teramoto, Tetsuya Kumagai, Tomohiro Tamura, Tomoshi Kameda, Wataru Kitagawa, and Tatsuya Tamei (2019)

[23] A critical analysis of codon optimization in human therapeutics by Nicholas J.

Proudfoot and Andrew D. Wipat (2016)

[24] Cynthia Gibas, Per Jambeck, Developing Bioinformatic Computer Skills, 2001,O'Rielly, 26-29

[25] Lynch, M. (2007). Evolution of the genetic code. Cold Spring Harbor Laboratory Press

[26] Pál, C., & Hurst, L. D. (2004). Human codon usage and the Neutrality Hypothesis. Human Genetics, 115(6), 505-511

[27] Tuller, T., & Hoekstra, R. F. (2007). Evolution of synonymous codon usage in Drosophila. Genetics, 176(3), 1431-1440

[28] Andersson, S. G. E., & Kurland, C. G. (1990). Codon usage in Escherichia coli: Optimization for translation efficiency. Journal of Molecular Biology, 218(4), 455-466

[29] Gustafsson, C., Goldman, N., & Fickett, J. W. (2004). Optimizing gene expression in Escherichia coli

[30] Inglis, S. C., et al. (2012). The impact of codon usage on heterologous protein expression in mammalian cells. Journal of Biotechnology, 157(3), 207-215

WEBSITES:

- www.ncbi.nlm.nih.gov
- www.wikipedia.com

10 APPENDIX A – Prototypes

The prototype of this project is the website: https://custom-cai.onrender.com/

- The Home page of our website emerges first:

Fig 15: Homepage of the website from a Desktop/Laptop

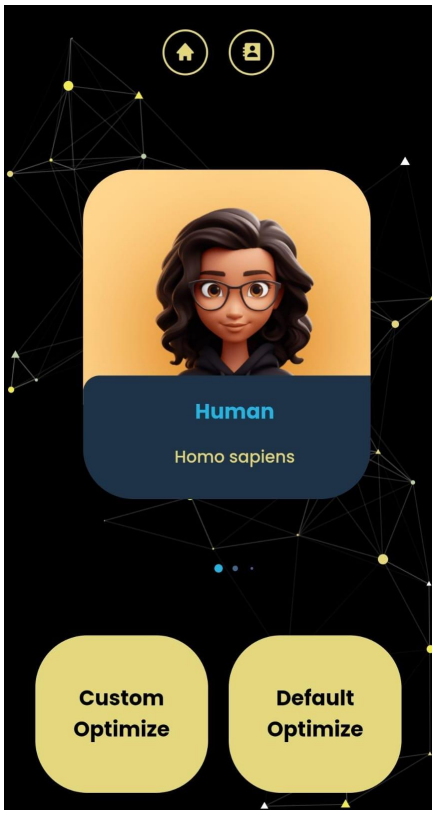

Fig 16: Homepage of the website from a Mobile

- We have two options to choose from: Custom Optimize and Default Optimize.

- Custom Optimize: Here the codon optimization will take place based on the codon usage table provided by the user in .csv format.
- Default Optimize: Here the codon optimization will work on the given codon usage table in the algorithm.

- Upon selecting Default Optimize option, we get a page as shown:

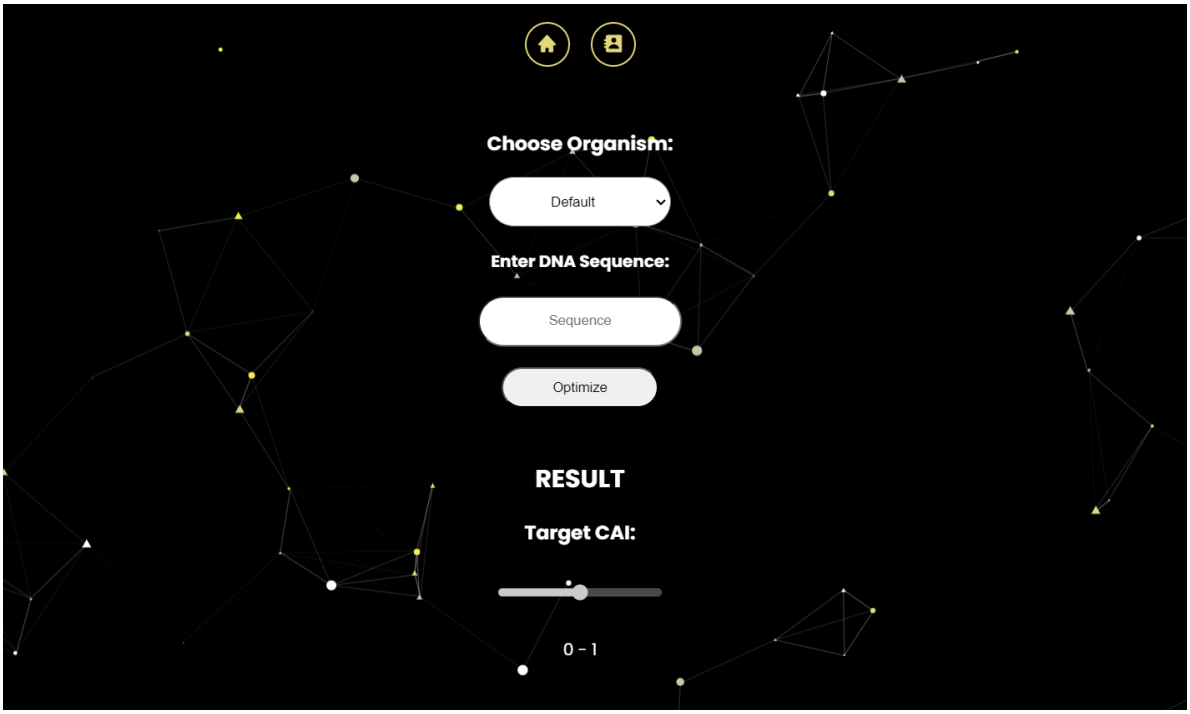

Fig 17: On selecting the Default Optimize option

- We can choose any organism from the given options. Then we need to enter the DNA sequence that we would want to optimize.
  As a result, we will get the optimized sequence along with the optimized CAI.

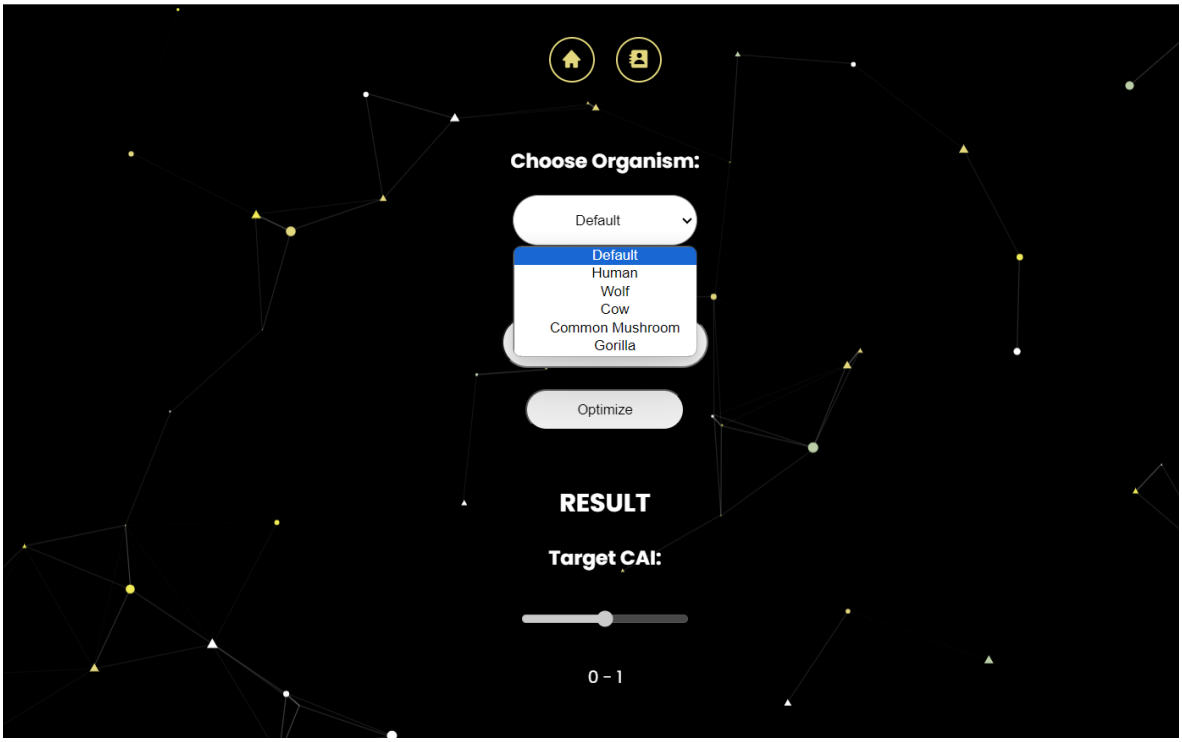

Fig 18 : The various organisms to select from

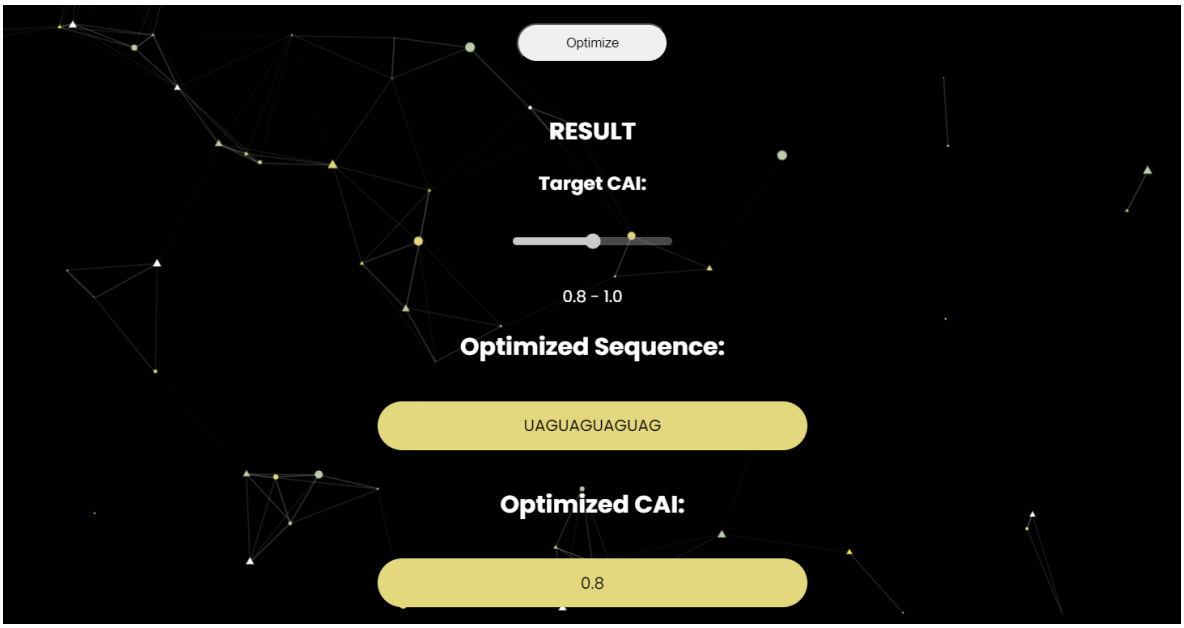

Fig 19: Result of the process

- Upon selecting Custom Optimize option, we get a page as shown:

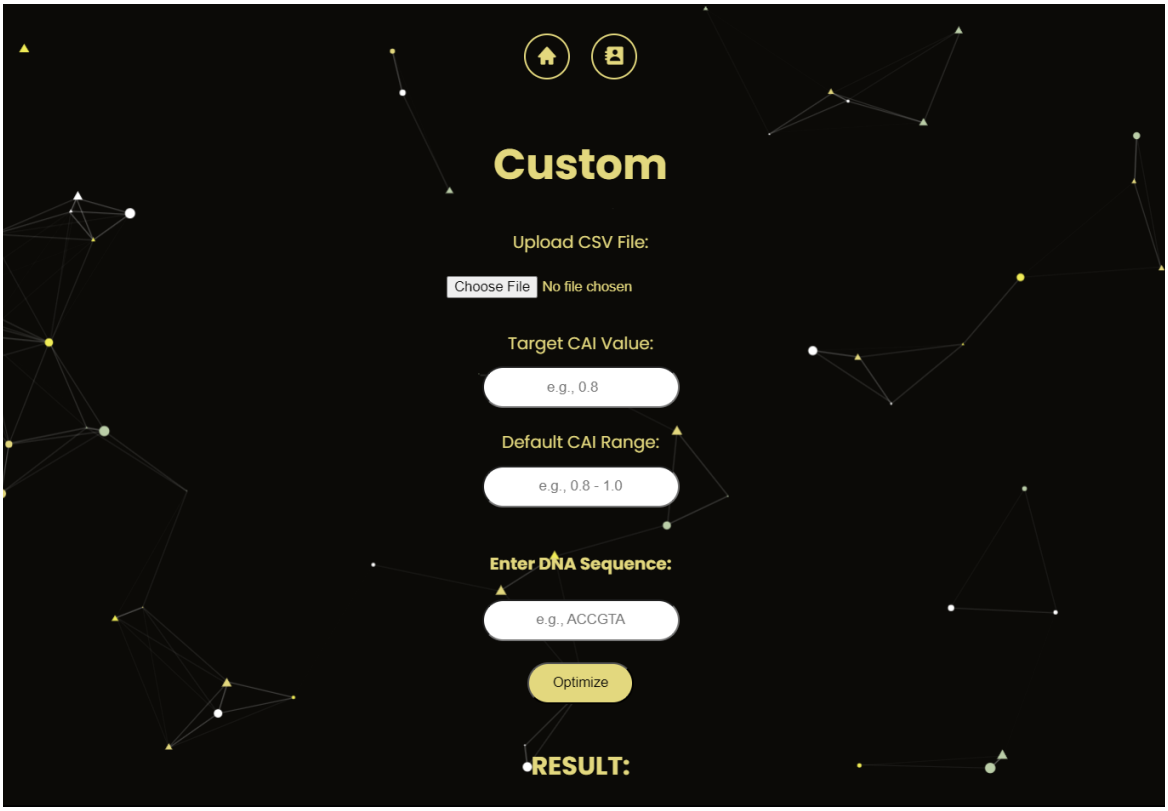

Fig 20: On selecting the Custom Optimize option

- We need to choose a .csv file to upload, which should contain the codon usage table we want. After that, the target CAI value, default CAI range and the DNA sequence, we want to optimize, needs to be entered.

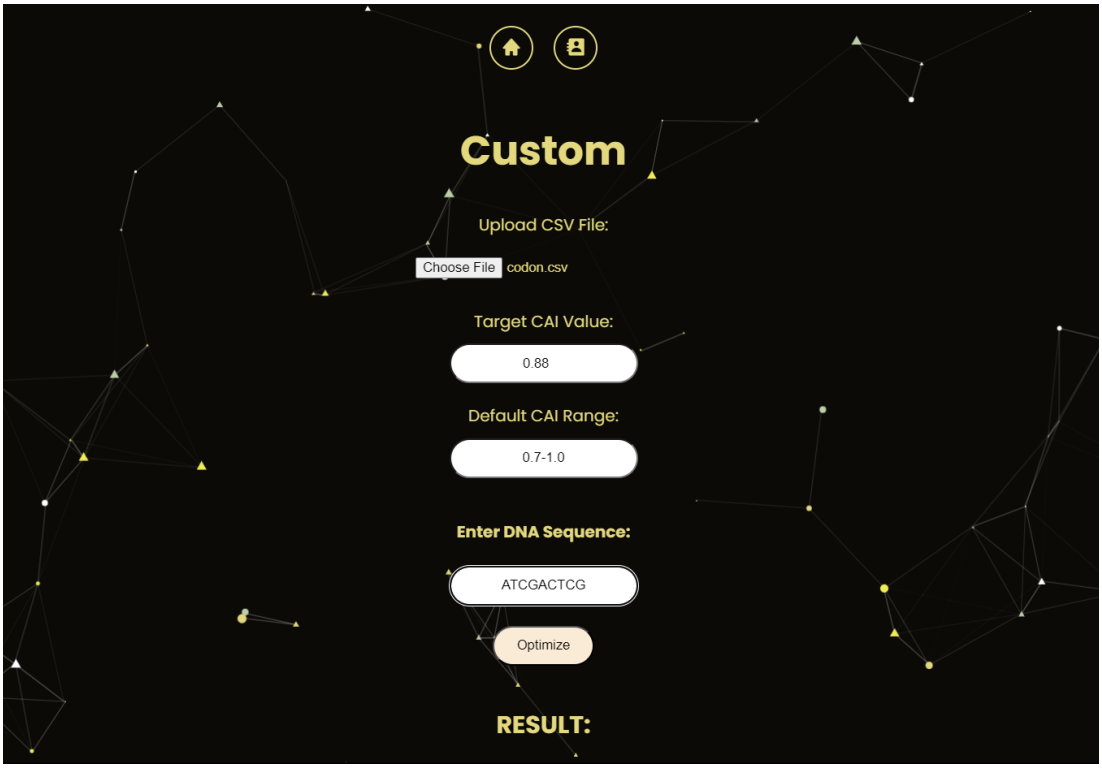

Fig 21: Filling up the details

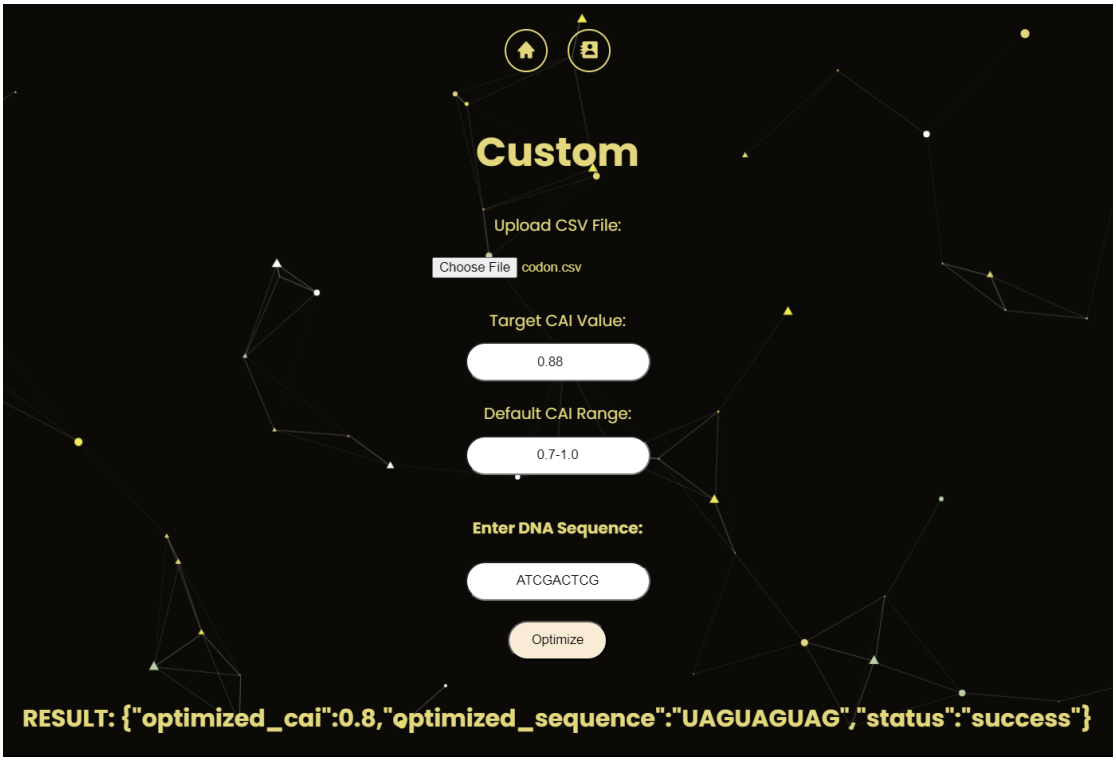

Fig 22: Result of the process

- If we choose a file in any format other than .csv, error will be displayed.

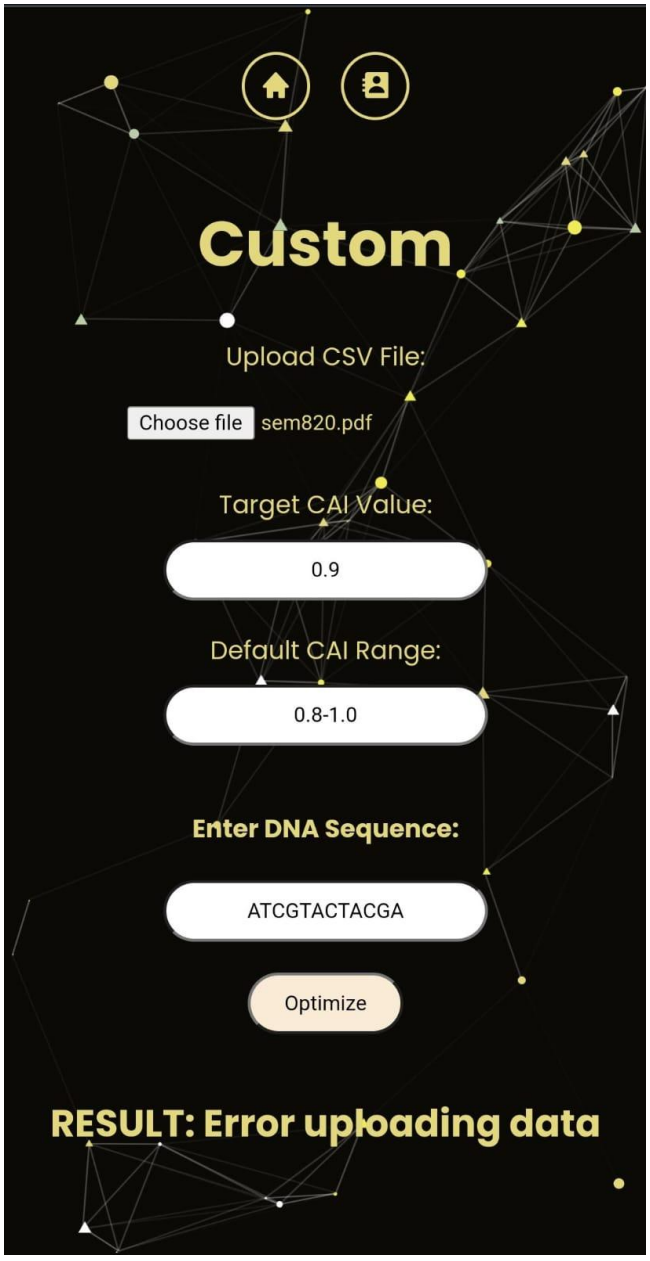

Fig 23: Error uploading data in wrong format

- On entering improper sequence to optimize, error will be shown.

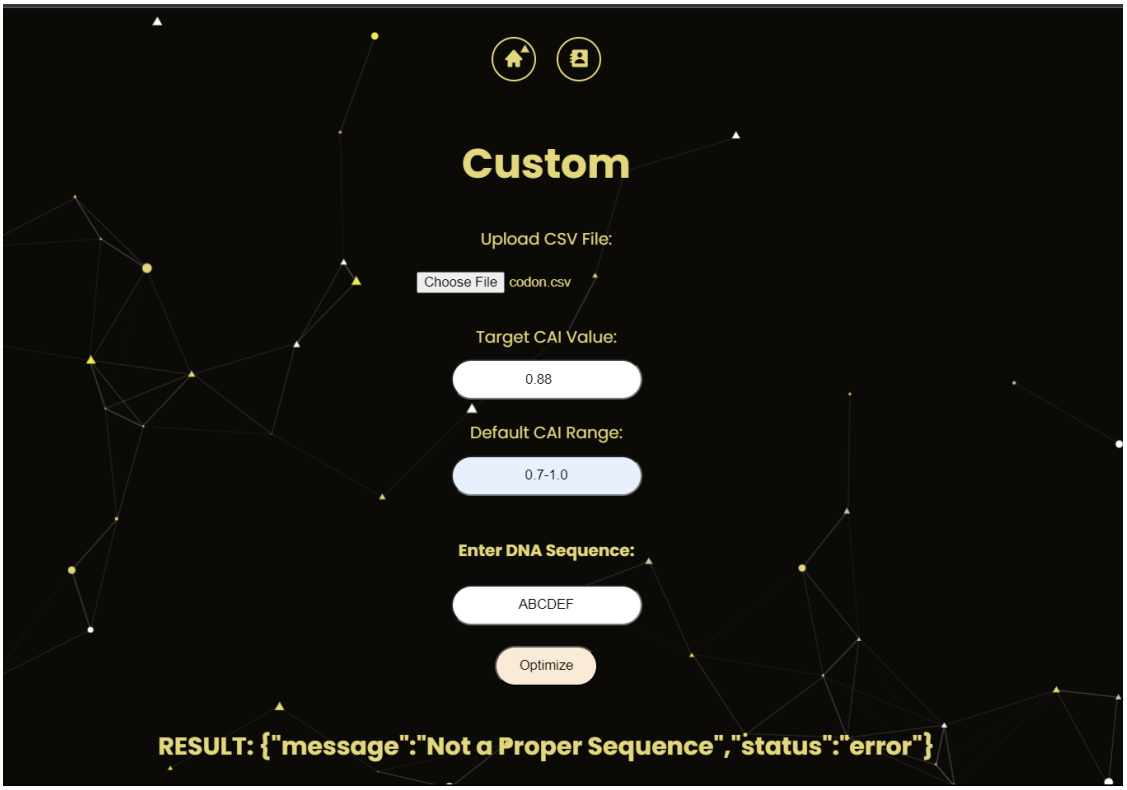

Fig 24: Improper sequence error