DNA Codec with Polar Codes, Huffman Compression and PGP Encryption

IT495 - DNA Storage

Team Members

Nilay Shah	201901026
Kathan Sanghavi	201901053
Aayush Prajapati	201901099
Yash Sakaria	201901165
Ronak Jethava	201901174
Parth Prajapati	201901429

Literature Review

Literature Review

- **Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels Erdal Arıkan, Senior Member, IEEE**

  This paper by Erdal Arıkan gives an idea about the construction of Error-correcting aka Capacity-achieving Polar Codes. We understood this paper's basic definition and implementation of Polar Encoder and Decoder. Based on the base formulas given in this, we are able to design efficient algorithms, that can be run in Python.

- **Introduction to Polar Codes, NPTEL - NOC IITM**

  This YouTube playlist is by Prof. Andrew Thangaraj. Through this playlist, he explains What are Polar Codes, Polar Transforms, Polar-Encoder and Polar-Decoder the help of an algorithm. We took reference of this video to get familiar with the concepts of Polar Transform, Polar Encoder and Polar Decoder for carrying out our research in DNA Codec.

- **A Brief Introduction to Polar Codes, Henry D. PfisterY.**

  This paper by Henry D. Pfister has basic details on kronecker delta products, designing efficient encoding and decoding algorithm and scheming polar code based on effective channel error rates. Also provides details on analysis on error rate of codes using monte carlo simulation of BSC, BEC channels.

Literature Review

- **Kurniawan, A. Albone and H. Rahyuwibowo, "The design of mini PGP security," Proceedings of the 2011 International Conference on Electrical Engineering and Informatics.**

  This paper gives an idea about creating a PGP application which has the conventional functions of encryption, decryption and digital signature. With help, we can plan to design the algorithm with some changes made in reference to considering the original system implementation.

- **“A Characterization of the DNA Data Storage Channel”, Reinhard Heckel1, Gediminas Mikutis2 & Robert N.Grass**

  This study describes the DNA data storage channel and how DNA can be utilised as an archival storage devices. It also underlines the limitations and imperfections in DNA synthesis, sequencing, and manipulation. We learned from others how vital it is to have a qualitative understanding of errors and molecule loss while designing a DNA storage system. In this paper, the error probabilities have also been described by analysing experimental data.

Encrypted Error Correcting - DNA Codec System

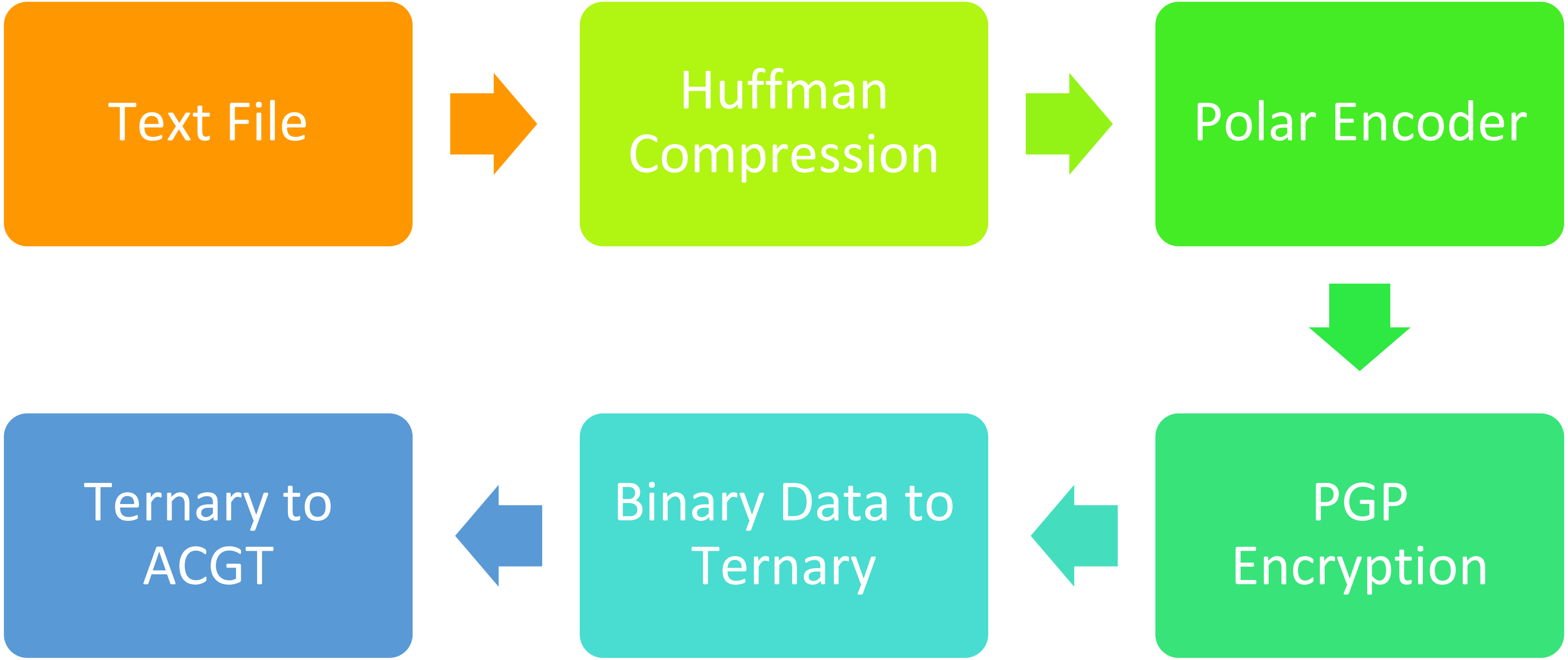
DNA Codec

Huffman Compression - Compressing initial size of data is the best choice for Archival Storage like DNA!

Polar Codes - Linear Block Error Correcting Code, going to be used in 5G! Will provide the capability of error correctness for our DNA storage.

PGP Encryption – Laptops/PCs are now coming with encrypted Hard Disks, so why not have similar security for DNA storage? PGP Encryption on DNA leverages the capability of DNA of storing huge amounts of data only using a few grams. We are using the GnuPG library of python to do encryption-decryption. Kleopatra software is used for key – certificate generation.

Encoder


Text File
Huffman Compression
Polar Encoder
PGP Encryption
Binary Data to Ternary
Ternary to ACGT

Text File:

Hello World!

Text File to Binary Data Conversion

ASCII to Binary: 01001000 01100101 01101100 01101100 01101111 00100000 01010111 01101111 01110010 01101100 01100100 00100001

Huffman Compression

Huffman Compression: 1100 1101 01 01 101 1110 1111 101 000 01 001 100

Polar Codes - Encoding

For Polar Codes Generator Matrix GN,

$$G_N = (I_{N/2} \otimes F)\, R_N\, (I_2 \otimes G_{N/2}), for\ N \geq 2$$

Where, $G_1 = I_1$. Using (1) we can get G_2 , G_3 , ... recursively.

We design an efficient encoding algorithm based on Generator Matrix.

Polar Encoding Calculation

$G_1 = [1]$

$$G_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

$$G_3 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

We get generator matrix using formula described. For coding part, we are using recursion and N can vary as we are not hardcoding the generator matrix.

Polar Encoding Pseudocode

```
def polar_encoder(u):
    if (len(u)==1):
        x = u
    else:
        u1u2 = (u[1::2]+u[2::2]) % 2
        u2 = u[2::2]
        x = [polar_encoder(u1u2), polar_encoder(u2)]
    return x
```

Polar Encoding

Polar Encoder, N = 7

1 1 0 1 1 0 0 0 1 1 1 0 0 1 0 0 1 0 0 0 0 0 1 0 0 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 1
0 1 1 1 1 1 0 1 1 0 1 1 0 0 0 1 1 1 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 1 0
1 1 1 0 0 0 1 1 1 0 0 0 1 0 1 0 0 0 0 1 0 0 1 1 0 1 1 1 1 0 0 1 1 1 0 1 0 0 1 1 1
0 0 0 1 0

PGP Encryption Block

PGP Encryption

01101000010001100011010001000100011000100011000001000101011
00110010000110101011001011010001100010101011101101110010100
01010100110100000101010001011001000100000101110100001100110
01101000011001001100010001100000101100000110000011110100101
01110111011101010101011110000111011001110001001100010101100
10110111101000000100110100010001100110110100110001001011101
10010001110011011011110110011001001110111100101110011010 10
00001010001011000110100111001010001010110010110010000101111
011000110100.....

Binary to Ternary Conversion

$$y_1 = x_1$$

$$y_i = x_i + x_{i-1} \; Where \; i = 2,3,\ldots,N$$

x(i-1) \ x(i)	0	1
0	0	1
1	1	2

Binary to Ternary

0121110001100121001211100110011001210011001210000110011112
1012101100012111111210111211100121001111111221121122101110
0111111012111000011111100111210110011000011122111000121012
0121110001210110121001100121000011121000012100001222110111
1122112211221111111111222100012211210122100110121001111210
1112112221110000110121110011001210121121110121001101112221
2101100122101211211222112101210110122211222101112210121111
0000111100111210012111012210111100111121011121011000111222
1121001211100121012111011121111211221122100011110121121000
2111001100110001211210012112211211112111000110121121110122
0001211112111011000111210001100121001210121121001211112101
0121001111011110110122101210001101111121121000111221011011
0012112211221121121001112111011012111012111001210011012210
2112221000110122101100001112210011110122101100001101211221
22211101211221001210121012101101111001111000121110122211 2..

Ternary to ACGT Conversion

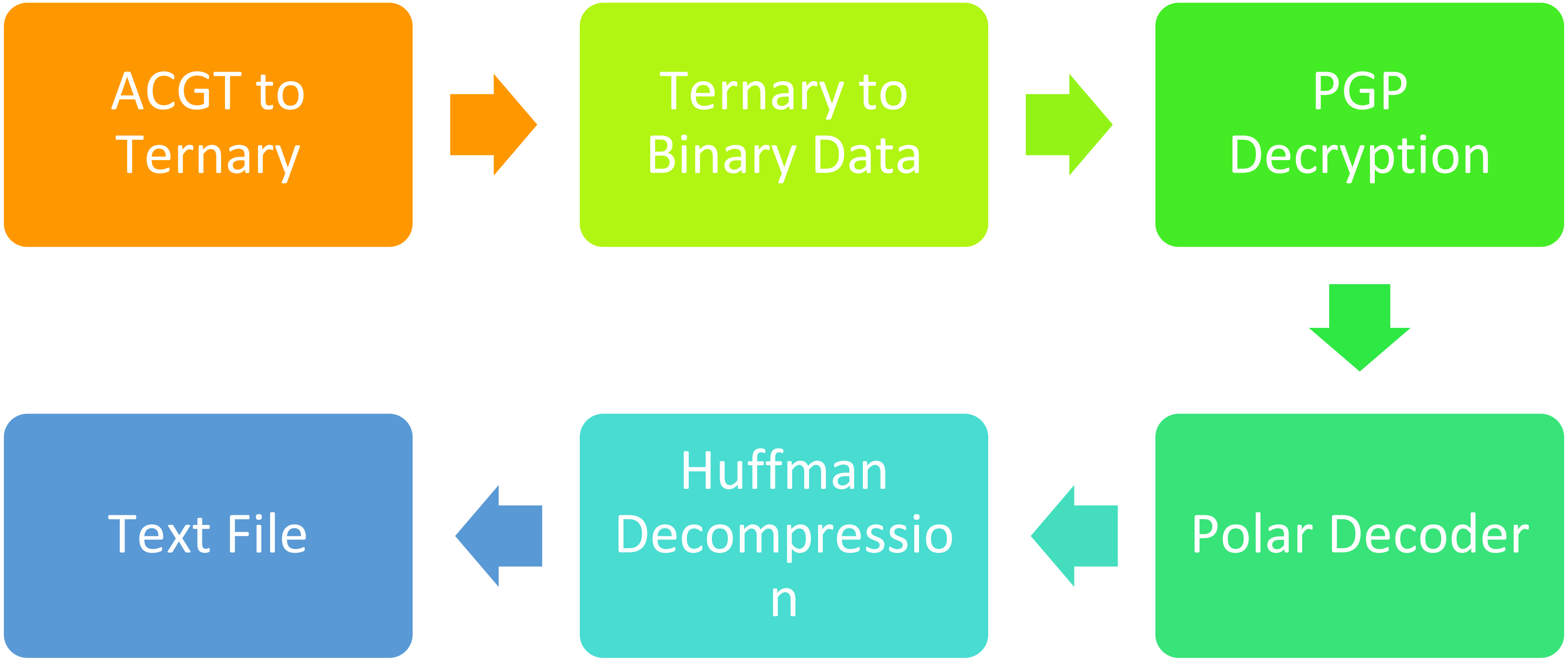
Mapping of y_1 $: 0 \rightarrow A,\ 1 \rightarrow C,\ \ 2 \rightarrow G$

y(i-1) \ x(i)	0	1	2
A	C	G	T
C	G	T	A
G	T	A	C
T	A	C	G

Ternary to ACGT

ACTGAGACGTCTACTGACGATCTCGTCTACTCGTCAGTAGACGATCGTACTCG
TCTCTCAGTCAGTCTACGATCTCTCTGACTCTGAGACGATCGTCTCTCTCATCT
GAGCAGTCTCTACTCTCTCGATCTCGTACTCTCTCGTCTCAGTCTACTCGTACT
CTGCTCTACGATCGATCGATCTCGTAGCTAGACTGACGAGTAGCTACGTCTCA
GTACGATCGTACTGCAGAGTCTCTCATCTGCTCATCTCTCTCTCTGCAGTACTG
CTCAGTCATCGTCTAGCTACTCTCTGACTCTGAGCATCTCGTACTCGATCTCGT
CTACTGACTGAGCTCTAGCTACTCGAGATGCTCAGTCTACTGCTAGCTCAGAT
GCTCAGTCAGTCTAGCATCTGCAGTCTCATCGATCTCTCGTACTCTCGTCTCAG
TAGCTCTAGCAGTCTCTACTCTCTGACTCTGACTCGTAGAGCATCTGACGATCT
CGTCAGTCAGAGTCTCAGAGATCTGCTCATCGTAGAGACTGAGCTACGATCTC
GTCTACTCGTAGCTCAGTAGCTCATCTGAGAGCTCTACGAGTCAGATCT..

Decoder


ACGT to Ternary
Ternary to Binary Data
PGP Decryption
Polar Decoder
Huffman Decompression
Text File

ACGT Data

ACTGAGACGTCTACTGACGATCTCGTCTACTCGTCAGTAGACGATCGTACTC
GTCTCTCAGTCAGTCTACGATCTCTCTGACTCTGAGACGATCGTCTCTCTCAT
CTGAGCAGTCTCTACTCTCTCGATCTCGTACTCTCTCGTCTCAGTCTACTCGT
ACTCTGCTCTACGATCGATCGATCTCGTAGCTAGACTGACGAGTAGCTACGT
CTCAGTACGATCGTACTGCAGAGTCTCTCATCTGCTCATCTCTCTCTCTGCAG
TACTGCTCAGTCATCGTCTAGCTACTCTCTGACTCTGAGCATCTCGTACTCGA
TCTCGTCTACTGACTGAGCTCTAGCTACTCGAGATGCTCAGTCTACTGCTAG
CTCAGATGCTCAGTCAGTCTAGCATCTGCAGTCTCATCGATCTCTCGTACTCT
CGTCTCAGTAGCTCTAGCAGTCTCTACTCTCTGACTCTGACTCGTAGAGCATC
TGACGATCTCGTCAGTCAGAGTCTCAGAGATCTGCTCATCGTAGAGACTGAG
CTACGATCTCGTCTACTCGTAGCTCAGTAGCTCATCTGAGAGCTCTACGAGT
CAGATCT..

ACGT to Ternary Conversion

$$\textit{Mapping of } y_1 : 0 \rightarrow A,\ 1 \rightarrow C,\ \ 2 \rightarrow G$$

y(i) / y(i-1)	A	C	G	T
A	-	0	1	2
C	0	-	1	2
G	0	1	-	2
T	0	1	2	-

ACGT to Ternary Conversion

0121110001100121001211100110011001210011001210000110011111210
1210110001211111121011121110012100111111122112112210111100111
1110121110000111111001112101100110000111221110001210121012111
0001210110121001100121000011121000012100001222111011111221122
1122111111111122210001221121012210011012100111112101112112221
1100001101211100110012101211211101210011011112221121011001221 0
1211211222112101210110122211222101112210121111100001111001112
1001211101221011110011111210111210110001112222112100121110012 1
0121110111211111211221122100011110121121000121110011001100012 1
1210012112211211112111000110121121110122100012111121110110001
1121000110012100121012112100121111210110121001111011110110122
1012100011011111211210001112210110110001211221122112112100111
2111011012111012111001210011012210121122210001101221011000011
1221001111012210110000110121122112221110121122100121012101210
1101111001111000121110122211 2..

Ternary to Binary Conversion

$$x_1 = y_1$$

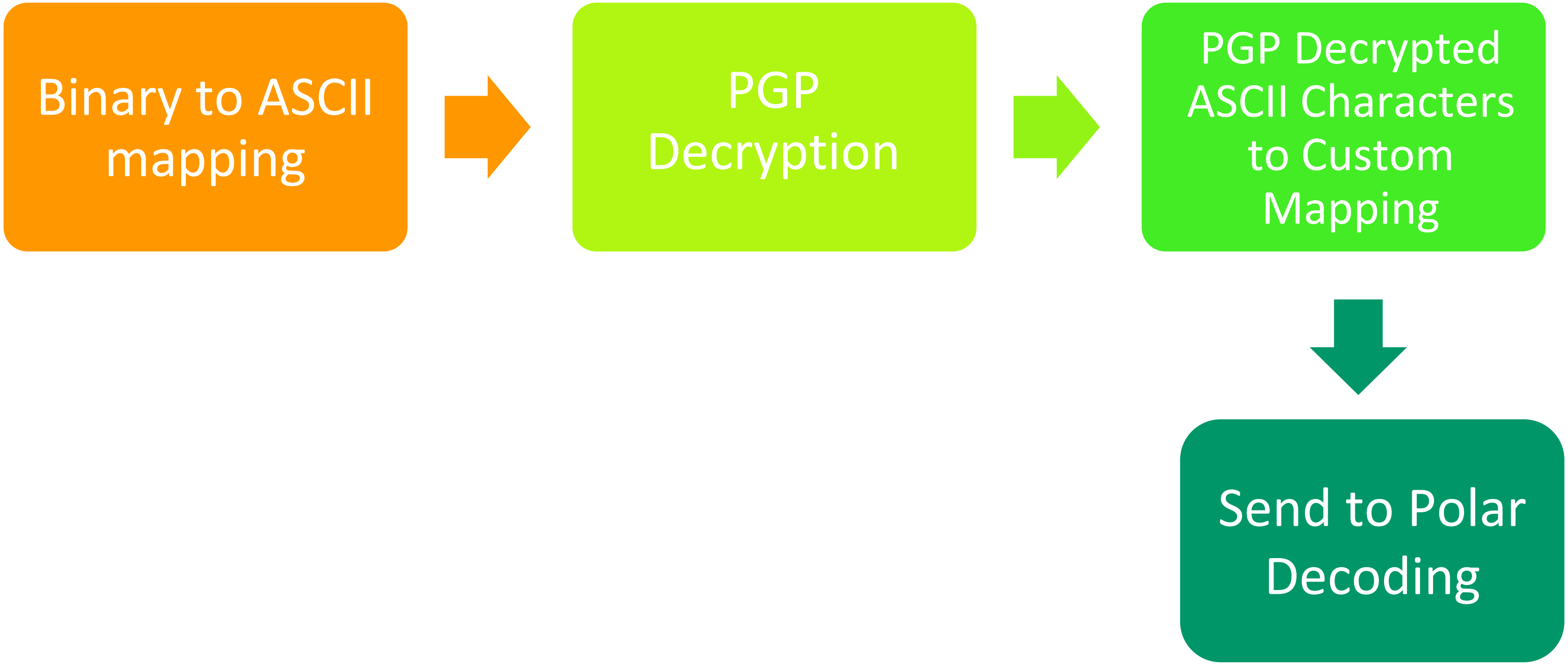
$$x_i = (y_i + x_{i-1}) mod\ 2, \qquad Where\ i = 2,3, \ldots, N$$

y(i) / x(i-1)	0	1	2
0	0	1	–
1	–	0	1

Ternary to Binary Conversion

0110100001000110001101000100010001100010001100000100010101
1001100100001101010110010110100011000101010111011011100101
0001010100110100000101010001011001000100000101110100001100
1100110100001100100110001000110000010110000011000001111010
0101011101110111010101010111100001110110011100010011000101
0110010110111101000001001101000100011001101101001100010010
1111011001000111001101101111011001100100111101111001011100
1101010000010100010110001101001110010100010101100101100100
0010111101100011 0100.....

PGP Decryption Block



PGP Decryption

1 1 0 1 1 0 0 0 1 1 1 0 0 1 0 0 1 0 0 0 0 0 1 0 0 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0
1 0 1 1 1 1 1 0 1 1 0 1 1 0 0 0 1 1 1 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0
1 0 1 1 1 0 0 0 1 1 1 0 0 0 1 0 1 0 0 0 0 1 0 0 1 1 0 1 1 1 1 0 0 1 1 1 0 1 0 0
1 1 1 0 0 0 1 0

Polar Codes - Decoding

Worst Case time complexity for decoding block-length N code is O(NlogN).

Decoder can be seen as N decision elements, one for each source element ui. They are activated

in the order 1 to N according to the likelihood ratio,

$$L_N^{(i)}\left(y_1^N, \hat{u}_1^{i-1}\right) \triangleq \frac{W_N^{(i)}\left(y_1^N, \hat{u}_1^{i-1}|0\right)}{W_N^{(i)}\left(y_1^N, \hat{u}_1^{i-1}|1\right)}$$

Polar Codes - Decoding

Decision at $\hat{u}_i$ is generated as,

$$\hat{u}_i = \begin{cases} 0, & if\ L_N^{(i)}\left(y_1^N, \hat{u}_1^{i-1}\right) \geq 1 \\ 1, & otherwise \end{cases}$$

Efficient Decoding Algorithm is made based on equation (1) and (2).

Polar Decoding

1100 1101 01 01 101 1110 1111 101 000 01 001 100

Huffman Decompression

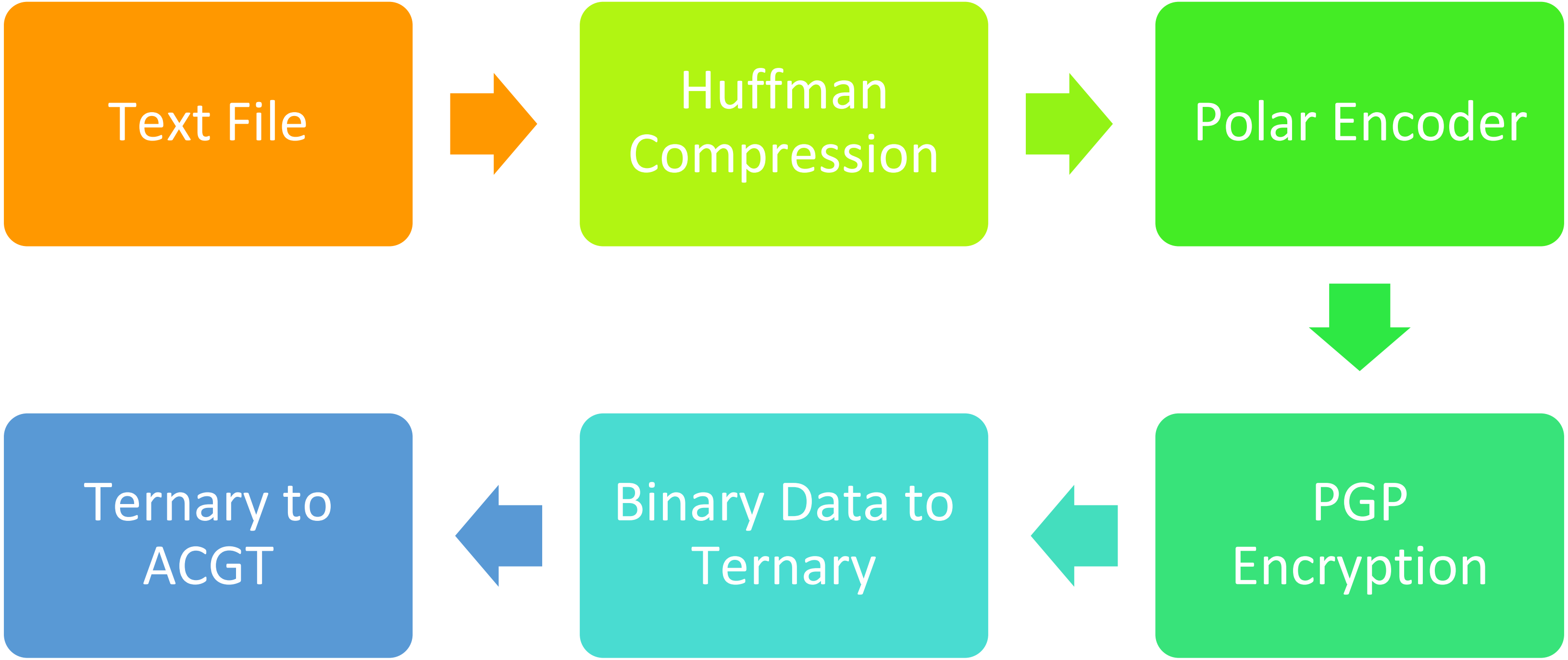
01001000 01100101 01101100 01101100 01101111 00100000
01010111 01101111 01110010 01101100 01100100 00100001

Binary to ASCII

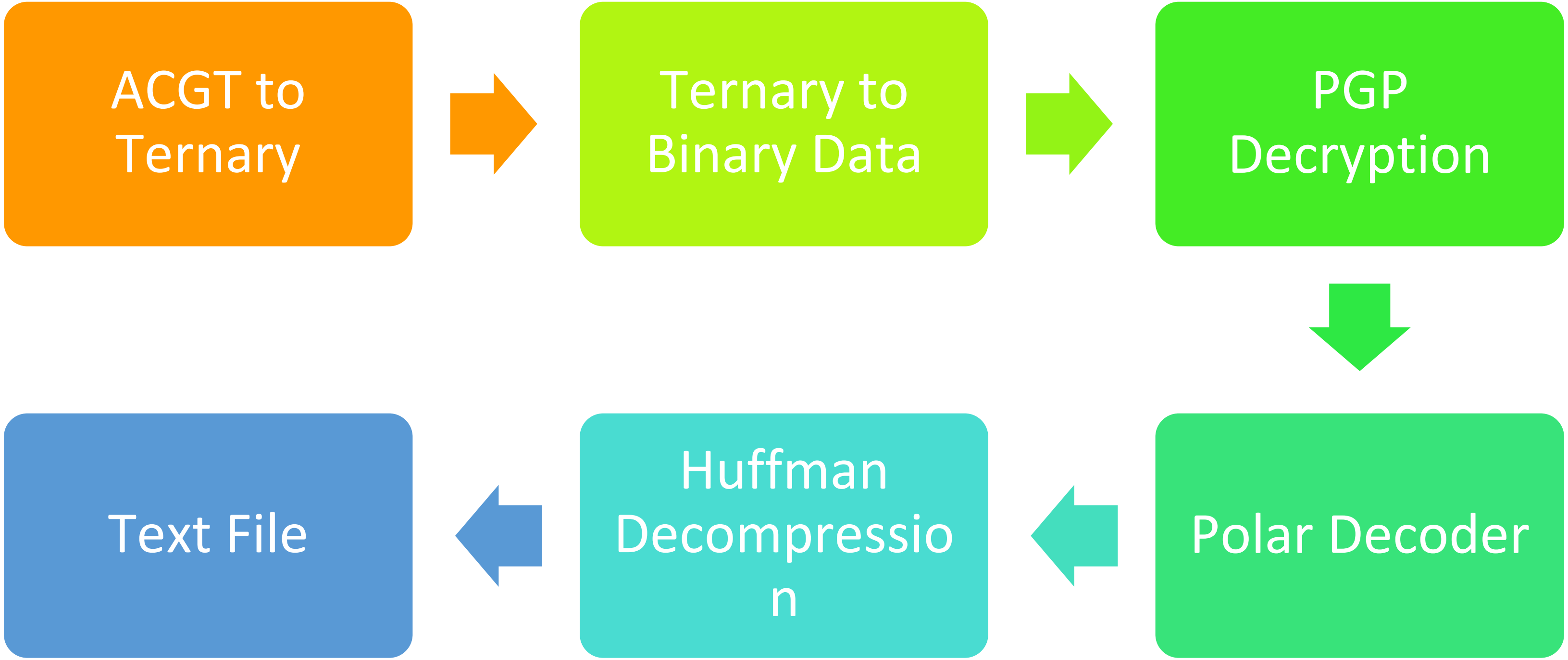
Hello World!

User Application - UI Walkthrough

Encoder


Text File
Huffman Compression
Polar Encoder
PGP Encryption
Binary Data to Ternary
Ternary to ACGT

Decoder


ACGT to Ternary
Ternary to Binary Data
PGP Decryption
Polar Decoder
Huffman Decompressio
n
Text File

Applications

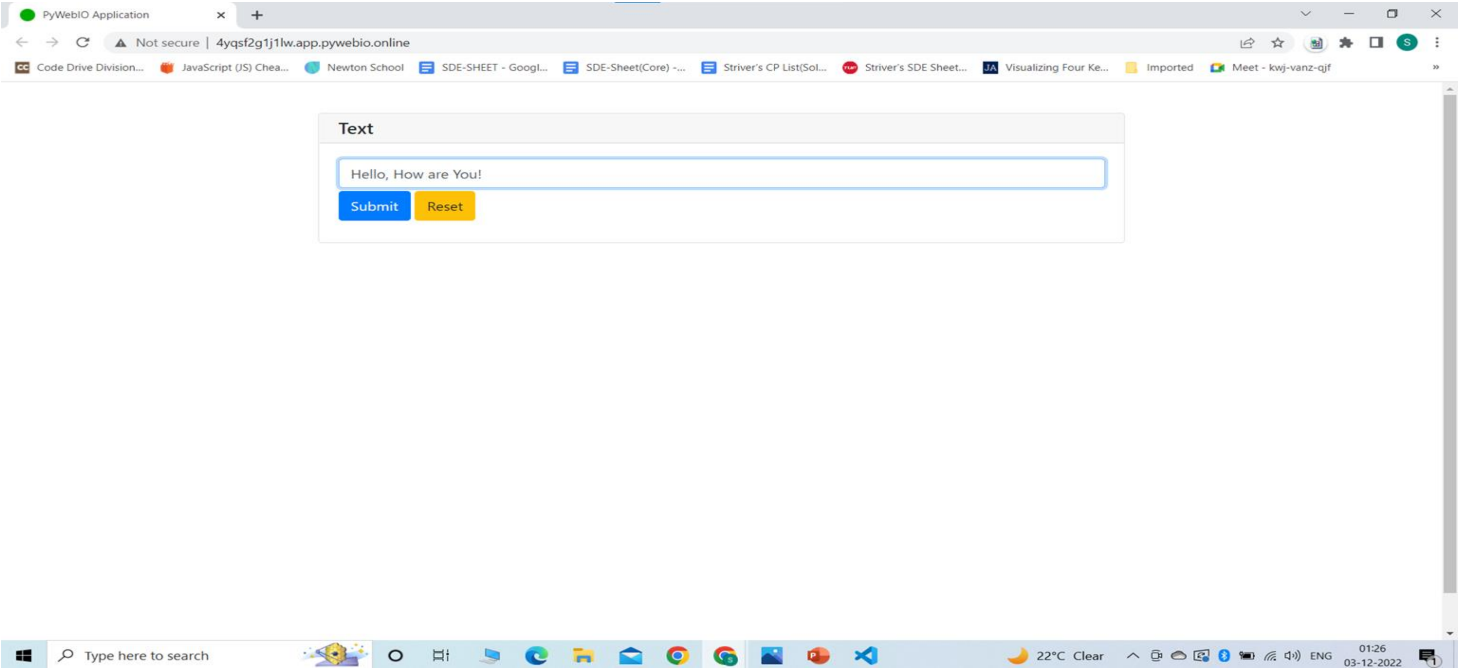
Encoder Web App: http://hn8qenyu4brw.app.pywebio.online/

Decoder Web App: http://hc2zz2pry0cj.app.pywebio.online/

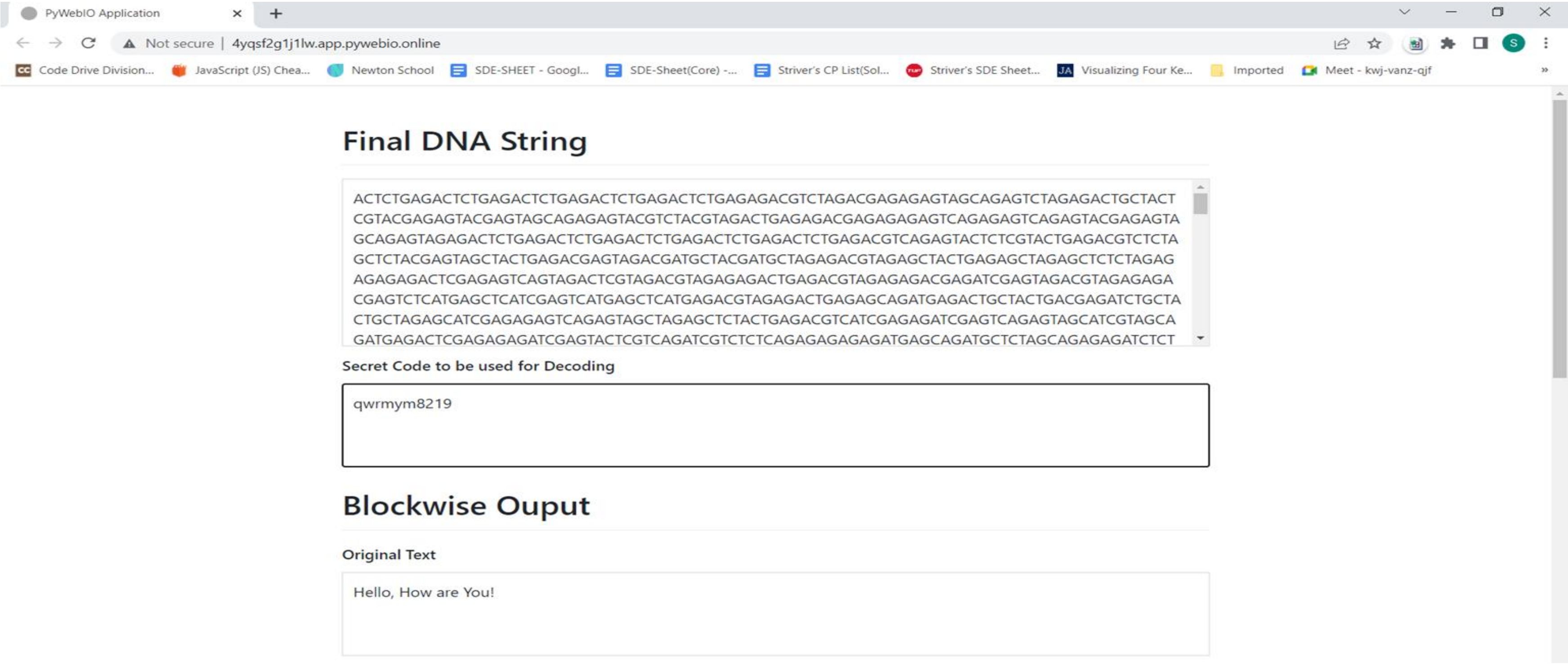
GitHub Repository: https://github.com/KathanS/DNA_Codec

User Application - System Design Overview

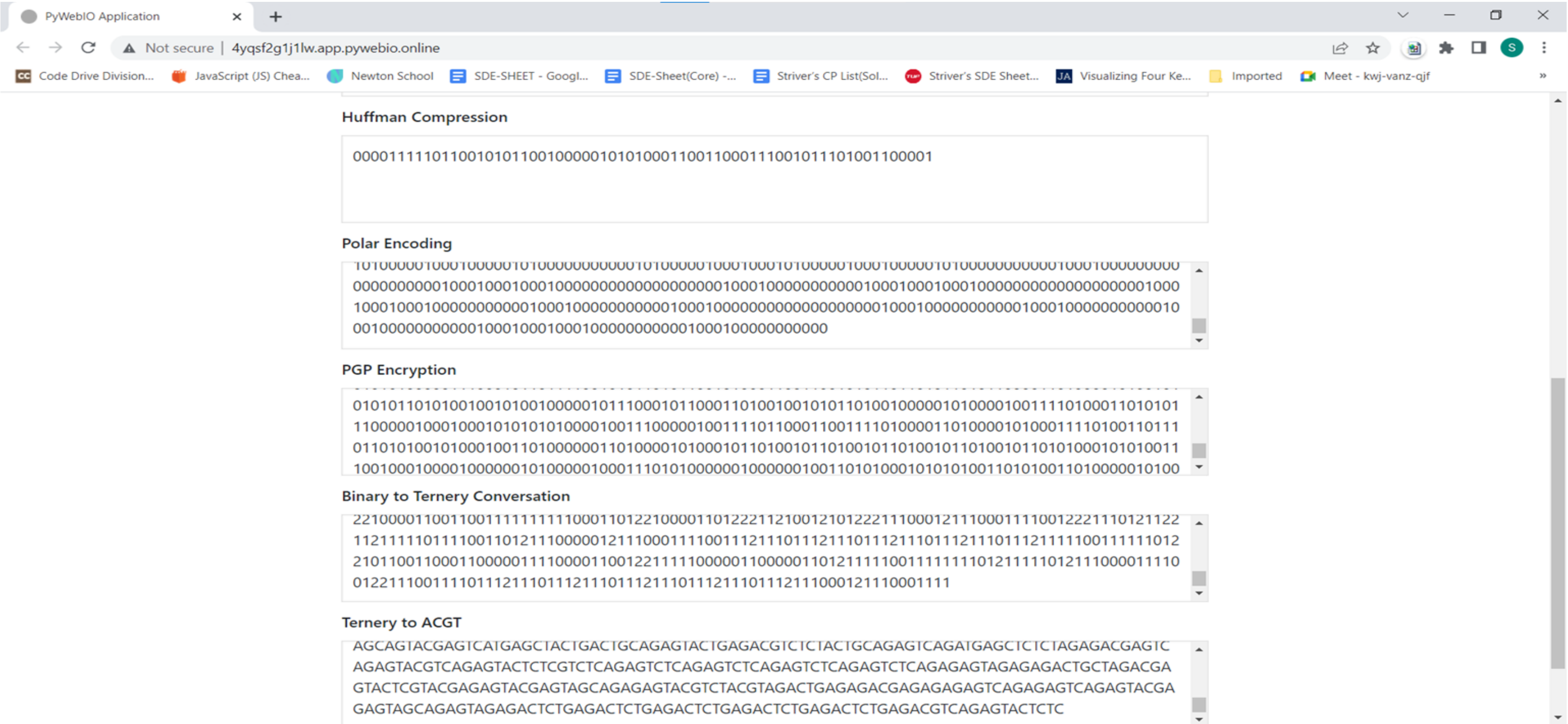
Encoding App Web interface



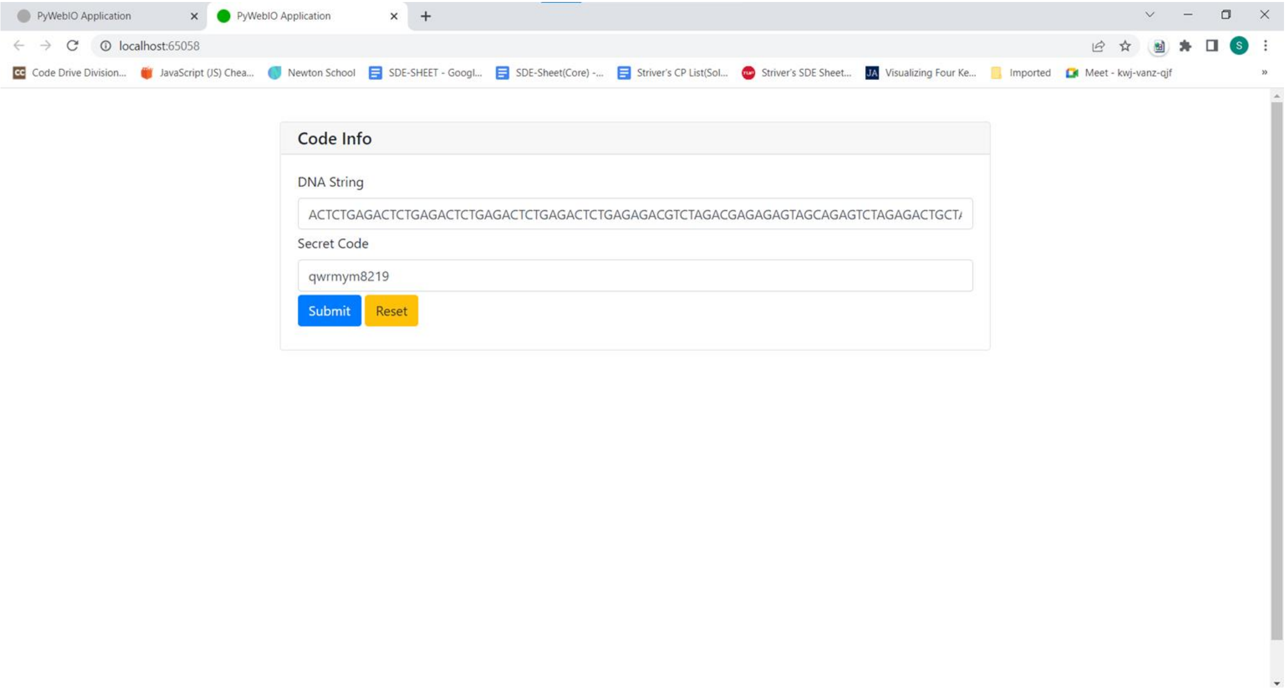
Blockwise Output [Encoder]



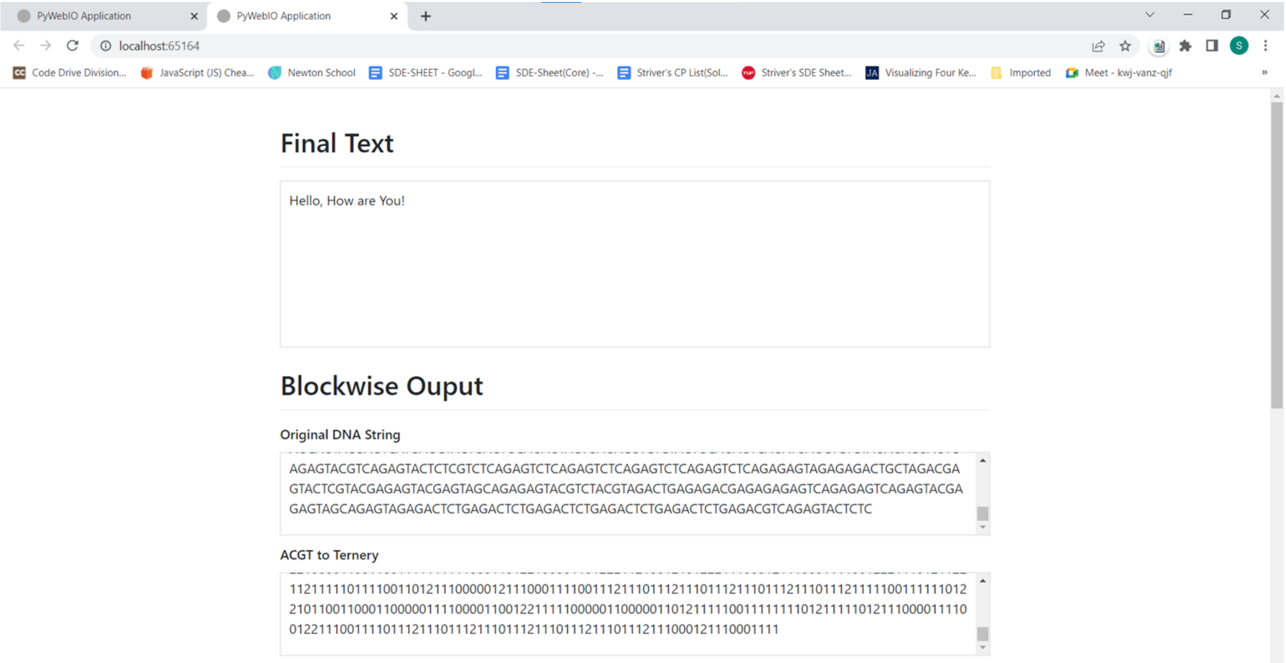
Blockwise Output [Encoder]



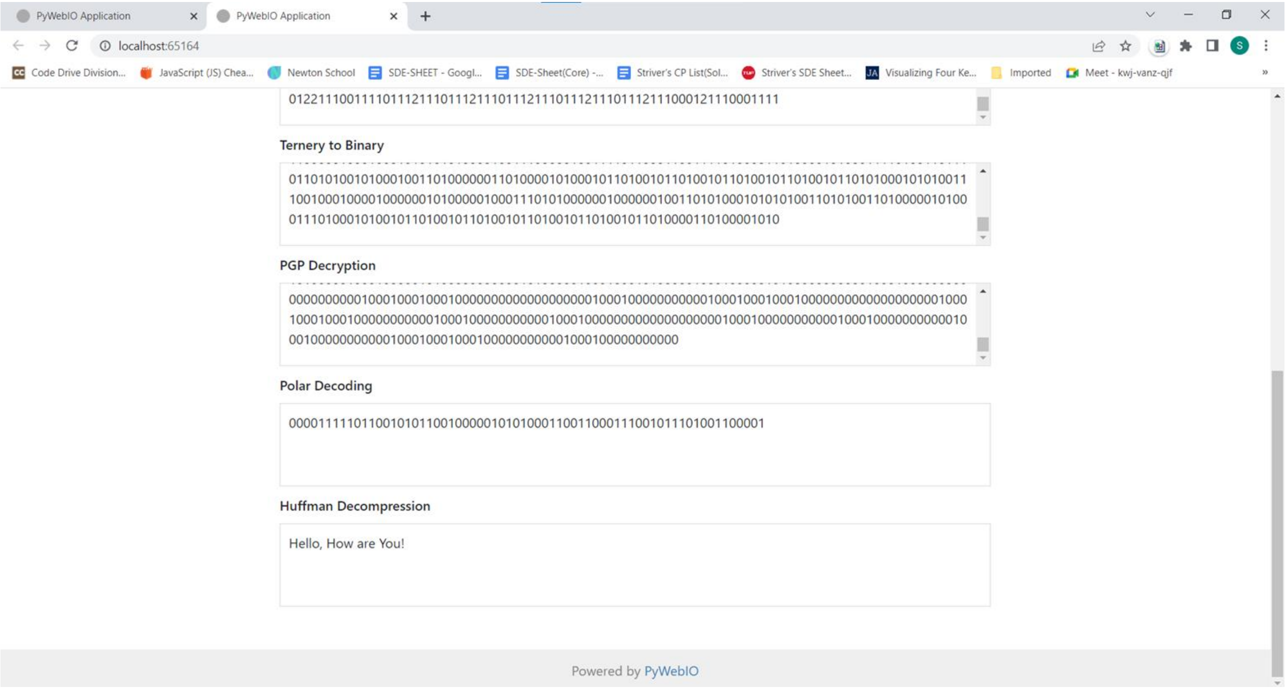
Decoding App Web Interface



Blockwise Output [Decoder]



Blockwise Output [Decoder]



Microservices like Architecture

- Encoder Web App outputs DNA string and Secret Code, which will be required for Decoding.

- We are using Python Pickles to store outputs of intermediate blocks, this way system gets decoupled and one block can pass the message having secret code to next block after finishing its task.

- We are able to scale each block independently, this increases robustness and overall performance of the system.

- **Information Density**

  Without PGP: ~2.5

  With PGP: ~6

References

1. Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels Erdal Arıkan, Senior Member, IEEE, https://arxiv.org/pdf/0807.3917.pdf
2. Introduction to Polar Codes, NPTEL - NOC IITM, (1225) Introduction to Polar Codes: Polar Transform - YouTube
3. A Brief Introduction to Polar Codes, Henry D. Pfister, polar.pdf (duke.edu)
4. A few concepts of DNA Codec - https://youtube.com/playlist?list=PLZ3QKRH1yB54D8HioxHVe7mrEx85yArOR
5. PGP Encryption/Decryption - What is PGP Encryption and How Does It Work? (varonis.com)
6. Huffman Compression - Huffman Encoding & Python Implementation | by Yağmur Çiğdem Aktaş | Towards Data Science
7. The design of mini PGP security, https://ieeexplore.ieee.org/document/6021726?arnumber=6021726