Assignment 3
Implementation of Vegenerce Cipher

PRN No. 2018BTECS00212

AIM: Implementation of Vegenerce Cipher.

THEORY:

Vigenerce cipher is a simple polyalphabetic cipher, in which the ciphertext is obtained by modular addition of a (repeating) key phrase and an open text (both of the same length).

Encryption:

$$C_i \equiv T_i + K_i \pmod{m}$$

Ci - i-th character of the ciphertext
Ti - i-th character of the open text
Ki - i-th character of the key phrase (if the key phrase is shorter than the open text, which is usual, than the keyphrase is repeated to math the length of the open text)
m - length of the alphabet

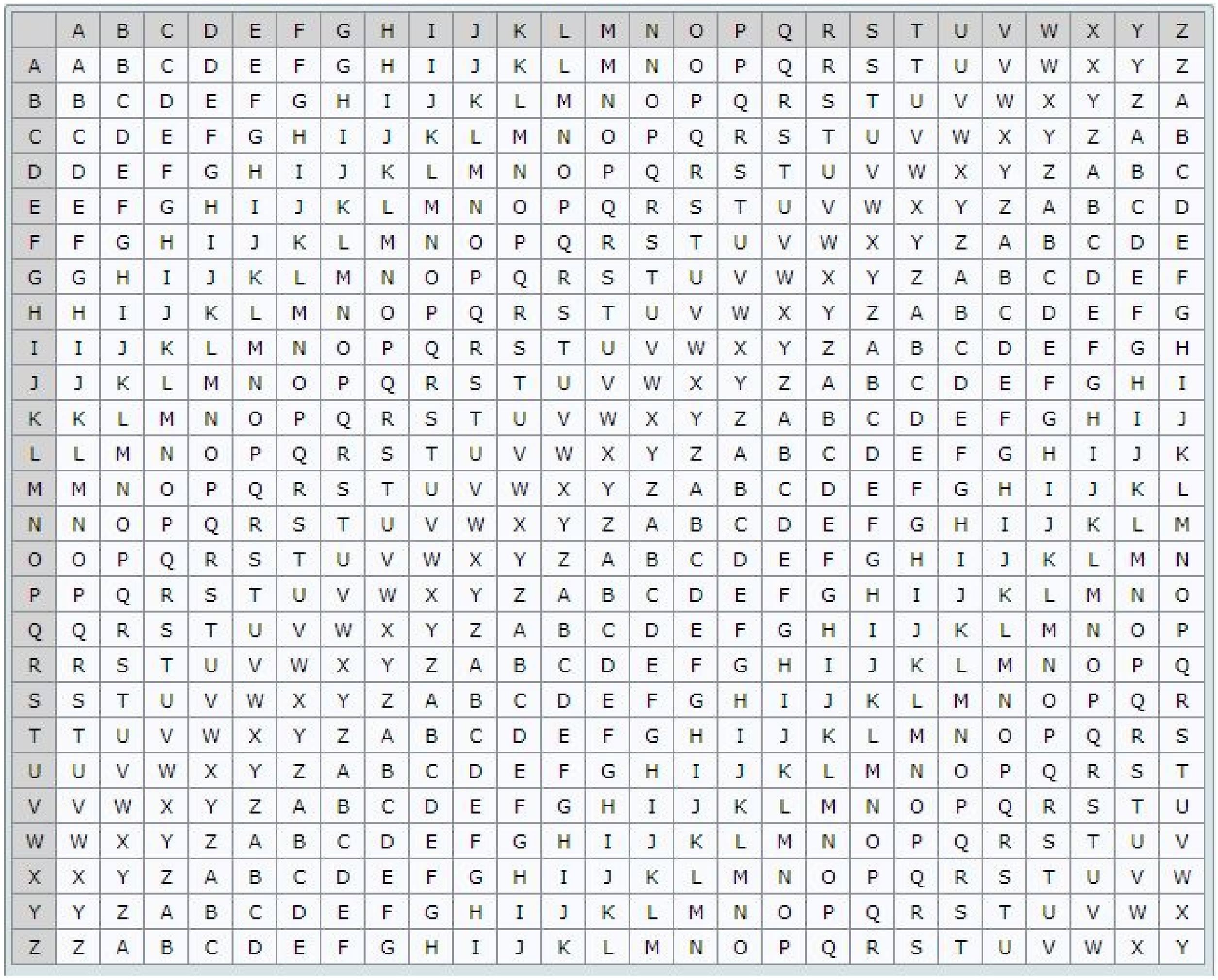
Decryption:

$$T_i \equiv C_i - K_i \pmod{m}$$

Ci - i-th character of the ciphertext
Ti - i-th character of the open text
Ki - i-th character of the key phrase (if the key phrase is shorter than the open text, which is usual, than the keyphrase is repeated to math the length of the open text)
m - length of the alphabet

PROCEDURE:

In order to simplify the encryption and decryption process, we may use Vigenère square (tabula recta). Each row of tabula recta consists of all letters of the English alphabet. The first row starts with the letter a, and each following row is shifted by one letter (second row starts with b, third with c...).



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

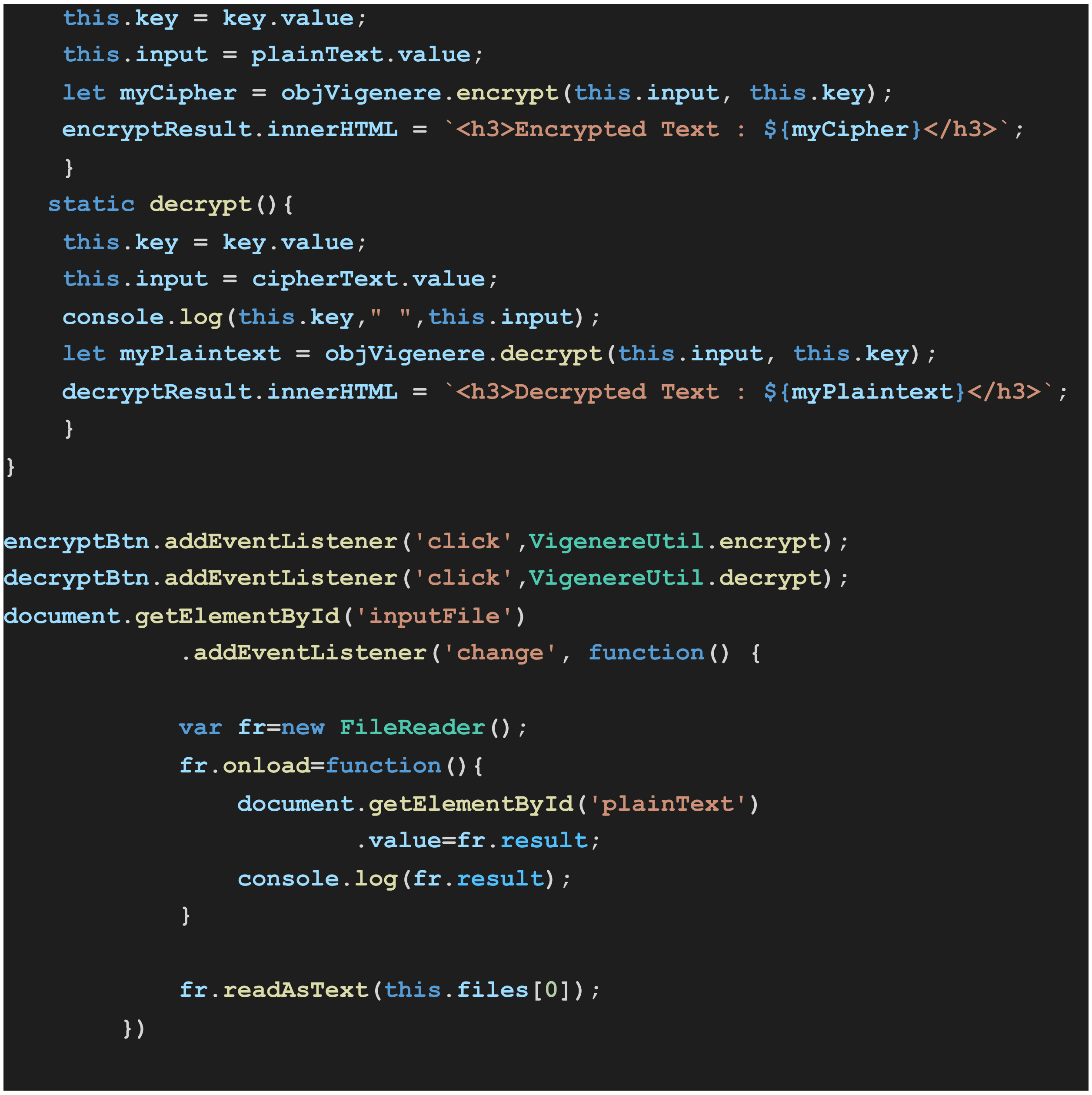
Processing of giving example:

Plaintext :	H	A	R	S	H	R	A	J
Code :	72	65	82	83	72	82	65	74
Key :	D	H	O	T	E	D	H	O
Code :	68	72	79	84	69	68	72	79
CipherText:	**K**	**H**	**F**	**L**	**L**	**U**	**H**	**X**

SOURCE CODE:

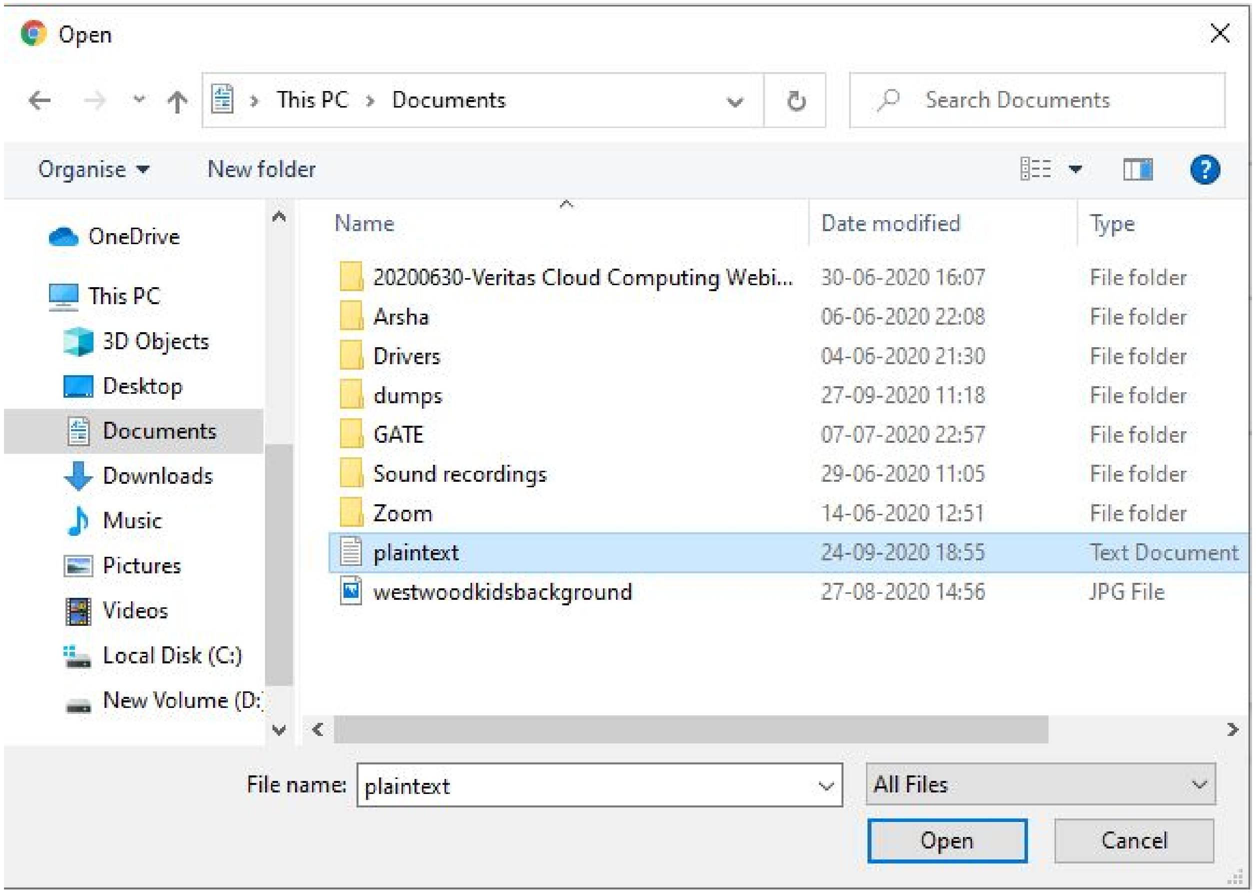
```
let plainText = document.getElementById('plainText');
let key = document.getElementById('key');
let decryptBtn = document.getElementById('decrypt-btn');
let encryptBtn = document.getElementById('encrypt-btn');
let encryptResult = document.getElementById('encryptResult');
let cipherText = document.getElementById('cipherText');
let decryptResult = document.getElementById('decryptResult');
```

```
function Vigenere() {
    "use strict";
    var plaintext = "";
    var ciphertext = "";
    var keyword = "";
    var alphabets = [];
    var init = function init() {

        var x;
        alphabets[0] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
        for (x = 1; x < alphabets[0].length; x = x + 1) {
            alphabets[x] = alphabets[0].substr(x);
            alphabets[x] = alphabets[x].concat(alphabets[0].substring(0,
x));
        }
    };

    var buildKeyword = function buildKeyword(password) {

        password = password.match(/[A-Za-z]/g);
        password = password.toString();
        password = password.replace(/[,]/g, "");

        keyword = password.toUpperCase();
    };

    this.encrypt = function encrypt(plaintext, password) {

        var x, pwIndex, vRow, thisLetter, thisRow;
         buildKeyword(password);
        plaintext = plaintext.toUpperCase();
        ciphertext = "";
        pwIndex = 0;
        for (x = 0; x < plaintext.length; x = x + 1) {
            vRow = alphabets[0].indexOf(keyword[pwIndex]);
            thisLetter = alphabets[0].indexOf(plaintext[x]);
            if (thisLetter === -1) {
                ciphertext += plaintext[x];
            } else {
                thisRow = alphabets[vRow];
                ciphertext += thisRow[thisLetter];
                pwIndex = pwIndex + 1;
            }
            if (pwIndex >= keyword.length) {
                pwIndex = 0;
            }
        }
        return ciphertext;
    };

    this.decrypt = function decrypt(ciphertext, password) {
        buildKeyword(password);
        plaintext = "";
        ciphertext = ciphertext.toUpperCase();

        var pwIndex, x, vRow, thisLetter, thisRow;
        pwIndex = 0;
        for (x = 0; x < ciphertext.length; x = x + 1) {
            vRow = alphabets[0].indexOf(keyword[pwIndex]);
            thisLetter = alphabets[vRow].indexOf(ciphertext[x]);
            if (thisLetter === -1) {
                plaintext += ciphertext[x];
            } else {
                thisRow = alphabets[0];
                plaintext += thisRow[thisLetter];
                pwIndex = pwIndex + 1;
            }
            if (pwIndex >= keyword.length) {
                pwIndex = 0;
            }
        }
        return plaintext;
    };
    init();
    return this;
}
var objVigenere = new Vigenere();
class VigenereUtil{

   static encrypt(){
```



```
    this.key = key.value;
    this.input = plainText.value;
    let myCipher = objVigenere.encrypt(this.input, this.key);
    encryptResult.innerHTML = `<h3>Encrypted Text : ${myCipher}</h3>`;
    }
   static decrypt(){
    this.key = key.value;
    this.input = cipherText.value;
    console.log(this.key," ",this.input);
    let myPlaintext = objVigenere.decrypt(this.input, this.key);
    decryptResult.innerHTML = `<h3>Decrypted Text : ${myPlaintext}</h3>`;
    }
}

encryptBtn.addEventListener('click',VigenereUtil.encrypt);
decryptBtn.addEventListener('click',VigenereUtil.decrypt);
document.getElementById('inputFile')
            .addEventListener('change', function() {

            var fr=new FileReader();
            fr.onload=function(){
                document.getElementById('plainText')
                        .value=fr.result;
                console.log(fr.result);
            }

            fr.readAsText(this.files[0]);
        })
```

O/P SCREENSHOTS :



Open

This PC › Documents

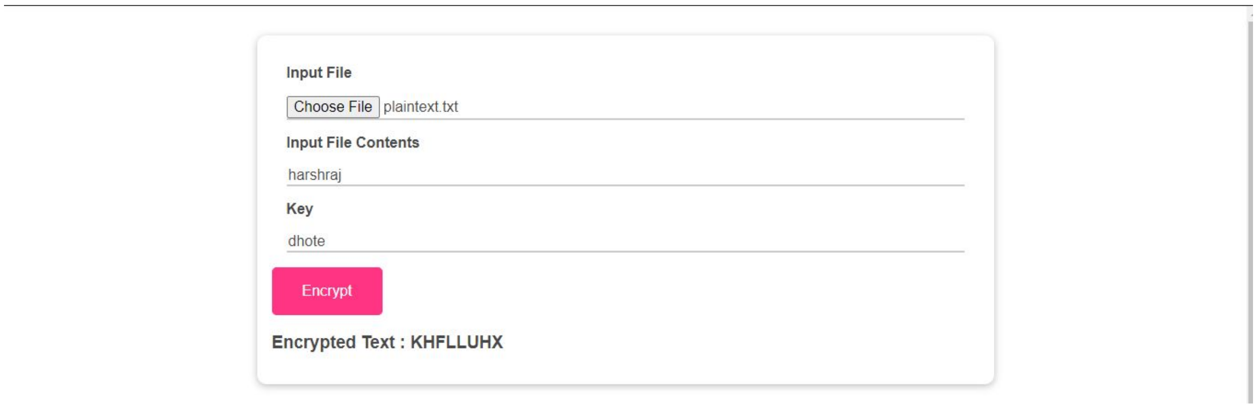
Search Documents

Organise ▾ New folder

Name	Date modified	Type
20200630-Veritas Cloud Computing Webi...	30-06-2020 16:07	File folder
Arsha	06-06-2020 22:08	File folder
Drivers	04-06-2020 21:30	File folder
dumps	27-09-2020 11:18	File folder
GATE	07-07-2020 22:57	File folder
Sound recordings	29-06-2020 11:05	File folder
Zoom	14-06-2020 12:51	File folder
plaintext	24-09-2020 18:55	Text Document
westwoodkidsbackground	27-08-2020 14:56	JPG File

File name: plaintext

All Files



Input File

Choose File plaintext.txt

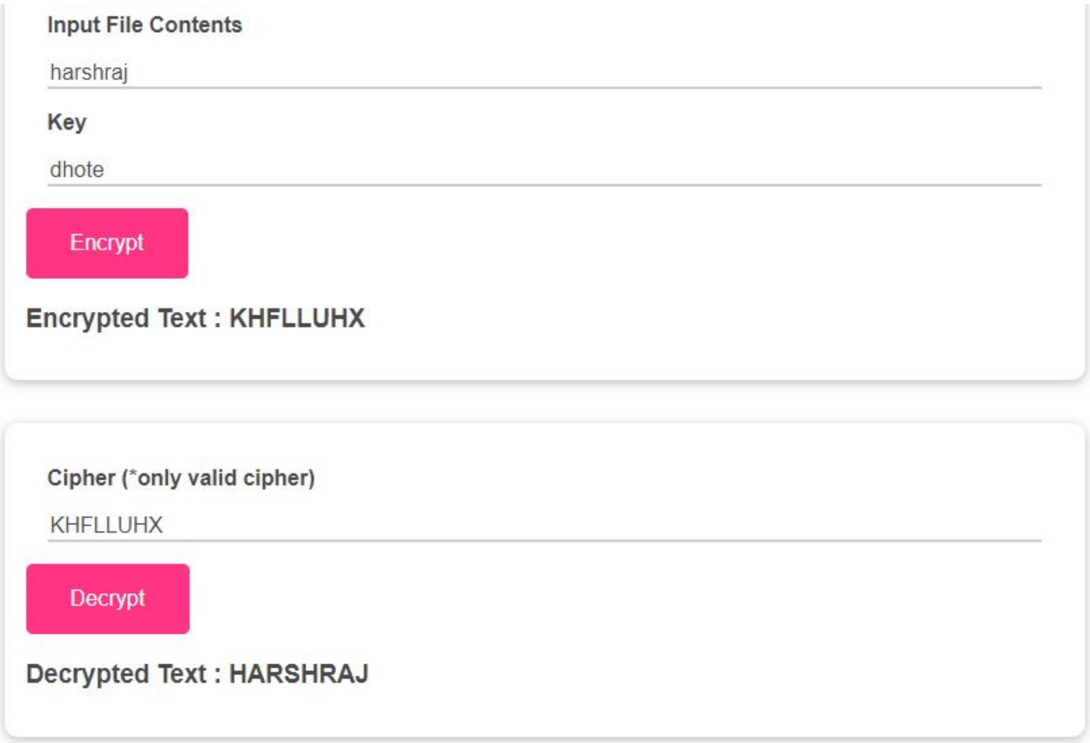
Input File Contents

harshraj

Key

dhote

Encrypted Text : KHFLLUHX


Input File Contents

harshraj

Key

dhote

Encrypted Text : KHFLLUHX

Cipher (*only valid cipher)

KHFLLUHX

Decrypted Text : HARSHRAJ

Conclusion:
Hence I successfully implemented the Vegenerce Cipher using Surname as a key for encryption and decryption.