# Laporan Tugas 1 IF4020 Kriptografi

## Semester II Tahun 2022/2023


Dibuat oleh :

13519174 - Jusuf Junior Athala

**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**2023**

## Bagian A

4. Link ke github atau google drive yang berisi kode program

A. Source program

1. File app.py

   File ini adalah file untuk menjalankan aplikasi dengan menggunakan Flask. File ini berisi fungsi-fungsi yang diperlukan untuk melakukan enkripsi dan dekripsi.

```python
from flask import Flask,render_template,request
import numpy as np


app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')


@app.route('/cryptography/', methods =["GET", "POST"])
def cryptography():
    if request.method == "POST":
        # getting input plaintext
        plaintext = request.form.get("plaintext")
       # getting input cyphertext
        cyphertext = request.form.get("cyphertext")
       # getting input method and key
        method = request.form.get("method")
        key = request.form.get("key")
        # getting input m and b for affine
        m = request.form.get("m")
        b = request.form.get("b")
        error_message = ""
```

```
        result_cyphertext=""
        result_decryptedtext=""

        # #get uploaded_file
        # plaintext_file = request.files["plaintext_file"]
        # cyphertext_file = request.files["cyphertext_file"]
        # file_contents = ''

        if (m=="" ):
            if ( method=='affine'):
                error_message +="m empty, using default m=7 for affine. "
                m= 7
            if (method=='hill'):
                error_message +="m empty, using default m=3 for hill. "
                m=3
        if (b=="" and method=='affine'):
            error_message +="b empty, using default b=10 for affine. "
            b= 10
        if request.form['submit_button'] == 'Encrypt!':
            result_cyphertext = encrypt(plaintext,key,method,m,b)
        if request.form['submit_button'] == 'Decrypt!':
            result_cyphertext = cyphertext
            result_decryptedtext = decrypt(cyphertext, key, method, m, b)
        return render_template("cryptography.html", plaintext=plaintext,
cyphertext=cyphertext,
                               method=method, key=key, m=m, b=b,
                               result_cyphertext = result_cyphertext,
                               result_decryptedtext = result_decryptedtext,
                               error_message = error_message )
    return render_template("cryptography.html")



def encrypt(plaintext, key, method, m, b):


    alphabets = 'abcdefghijklmnopqrstuvwxyz'
    cyphertext = ''
```

```
    len_key = len(key)
    if (method =='vigenere'):
        #     a) Vigenere Cipher standard (26 huruf alfabet)
        plaintext = plaintext.replace(" ","")
        plaintext = plaintext.lower()

        for i , char in enumerate(plaintext):
            p_val = alphabets.index(char)
            k_char = key[i % len_key]
            k_val = alphabets.index(k_char)
            c_val = (p_val + k_val) % 26
            cyphertext += str(alphabets[c_val])
    elif (method == 'auto-vigenere'):

        plaintext = plaintext.replace(" ","")
        plaintext = plaintext.lower()
         # b) Varian Vigenere Cipher (26 huruf alfabet): Auto-key Vigenere
Cipher
        auto_key = key+plaintext
        for i , char in enumerate(plaintext):
            p_val = alphabets.index(char)
            k_char = auto_key[i]
            k_val = alphabets.index(k_char)
            c_val = (p_val + k_val) % 26
            cyphertext += alphabets[c_val]
    elif (method == 'extended-vigenere'):
        # c) Extended Vigenere Cipher (256 karakter ASCII)
        ascii_chars = [chr(i) for i in range(256)]
        for i , char in enumerate(plaintext):
            p_val = ascii_chars.index(char)
            k_char = key[i % len_key]
            k_val = ascii_chars.index(k_char)
            c_val = (p_val + k_val) % 256
            cyphertext += str(ascii_chars[c_val])
    elif (method == 'affine'):

        plaintext = plaintext.replace(" ","")
        plaintext = plaintext.lower()
```

```
        m=int(m)
        b=int(b)
        # d) Affine Cipher
        for i , char in enumerate(plaintext):
            p_val = alphabets.index(char)
            c_val = ((m*p_val) + b) % len(alphabets)
            cyphertext += alphabets[c_val]
    elif (method == 'playfair'):

        plaintext = plaintext.replace(" ","")
        plaintext = plaintext.lower()
    # e) Playfair Cipher (26 huruf alfabet)
        #membuat grid playfair
        temp_key = key + alphabets
        temp_key = temp_key.replace("j","")
        playfair_key = ""
        for char in temp_key:
            if char not in playfair_key:
                playfair_key+=char
        grid =[ playfair_key[0:5],
               playfair_key[5:10],
               playfair_key[10:15],
               playfair_key[15:20],
               playfair_key[20:25]]

        #menyiapkan plaintext
        plaintext = plaintext.replace("j","")
        for i in range(0, len(plaintext)-1):
            p1= plaintext[i]
            p2= plaintext[i+1]
            if p1==p2:
                plaintext =  plaintext[:i+1] + 'x' + plaintext[i+1:]

        if (len(plaintext)%2==1):
                plaintext+='x'
        #menggeser plaintext berdasarkan matrix key
        for i in range(0, len(plaintext), 2):
```

```
            p1= plaintext[i]
            p2= plaintext[i+1]
            x1,y1 = getRowCol2d(grid,p1)
            x2,y2 = getRowCol2d(grid,p2)
            print(x1,y1,x2,y2)
            if x1==x2:
                y1= (y1+1)%5
                y2= (y2+1)%5
            elif y1==y2:
                x1= (x1+1)%5
                x2= (x2+1)%5
            else:
                temp = y1
                y1= y2
                y2= temp
            cyphertext+=grid[x1][y1]+grid[x2][y2]

    elif (method == 'hill'):
        app.logger.info('Hill Method')
        plaintext = plaintext.replace(" ","")
        plaintext = plaintext.lower()
    # f) Hill Cipher
        #membuat matrix key
        m=int(m)
        temp_key=''
        if (len_key<m**2):
            for i in range(0,m**2):
                temp_key+= key[i%len_key]
        if (len_key>=m**2):
            temp_key = key[:m**2]
        hill_key = []
        for i in range(0,len(temp_key),m):
            temp_arr=[]
            for j in range(m):
                k_char = temp_key[i+j]
                k_val = alphabets.index(k_char)
                temp_arr.append(k_val)
            hill_key.append(temp_arr)
```

```
        #membuat panjang plaintext habis dibagi oleh m
        #menambahkan sejumlah huruf x dibelakang agar panjang plaintext
        #habis dibagi oleh m
        mod_plaintext=len(plaintext)%m
        if (mod_plaintext!=0):
            plaintext+='x'*mod_plaintext

        #perkalian matrix hill cypher
        for i in range(0,len(plaintext),m):
            p_arr = []
            for j in range (m):
                p_val= alphabets.index(plaintext[i+j])
                p_arr.append(p_val)
            for row in range(m):
                c_val=0
                for col in range(m):
                    c_val += hill_key[row][col] *p_arr[col]
                cyphertext+= alphabets[c_val%26]


    return cyphertext


def getRowCol2d(arr_of_string,value):
    row=0
    col=0
    for word in arr_of_string:
        col=0
        for char in word:
            if char==value:
                return row,col
            col+=1
        row+=1
    return row,col
```

```
def decrypt(cyphertext, key, method, m, b):


    alphabets = 'abcdefghijklmnopqrstuvwxyz'
    decryptedtext = ''
    len_key = len(key)
    if (method =='vigenere'):
        #     a) Vigenere Cipher standard (26 huruf alfabet)

        cyphertext = cyphertext.replace(" ","")
        cyphertext = cyphertext.lower()


        for i , char in enumerate(cyphertext):

            c_val = alphabets.index(char)
            k_char = key[i % len_key]
            k_val = alphabets.index(k_char)
            p_val = (c_val - k_val) % 26
            decryptedtext += str(alphabets[p_val])
    elif (method == 'auto-vigenere'):
         # b) Varian Vigenere Cipher (26 huruf alfabet): Auto-key Vigenere
Cipher
        cyphertext = cyphertext.replace(" ","")
        cyphertext = cyphertext.lower()


        auto_key = key
        for i , char in enumerate(cyphertext):
            c_val=alphabets.index(char)
            k_char= auto_key[i]
            k_val = alphabets.index(k_char)
            p_val = (c_val - k_val) % 26
            decryptedtext += str(alphabets[p_val])
            auto_key += str(alphabets[p_val])
    elif (method == 'extended-vigenere'):
        # c) Extended Vigenere Cipher (256 karakter ASCII)
```

```
        ascii_chars = [chr(i) for i in range(256)]
        for i , char in enumerate(cyphertext):
            c_val = ascii_chars.index(char)
            k_char = key[i % len_key]
            k_val = ascii_chars.index(k_char)
            p_val = (c_val - k_val) % 256
            decryptedtext += str(ascii_chars[p_val])
    elif (method == 'affine'):

        cyphertext = cyphertext.replace(" ","")
        cyphertext = cyphertext.lower()



        m=int(m)
        b=int(b)
        m_inv= pow(m, -1, len(alphabets))
        # d) Affine Cipher
        for i , char in enumerate(cyphertext):
            c_val = alphabets.index(char)
            p_val = ((m_inv*(c_val-b)) )
            decryptedtext += alphabets[p_val % len(alphabets)]
    elif (method == 'playfair'):
        cyphertext = cyphertext.replace(" ","")
        cyphertext = cyphertext.lower()

    # e) Playfair Cipher (26 huruf alfabet)
        #membuat grid playfair
        temp_key = key + alphabets
        temp_key = temp_key.replace("j","")
        playfair_key = ""
        for char in temp_key:
            if char not in playfair_key:
                playfair_key+=char
        grid =[ playfair_key[0:5],
               playfair_key[5:10],
               playfair_key[10:15],
               playfair_key[15:20],
               playfair_key[20:25]]
```

```
        for i in range(0, len(cyphertext), 2):
            p1= cyphertext[i]
            p2= cyphertext[i+1]
            x1,y1 = getRowCol2d(grid,p1)
            x2,y2 = getRowCol2d(grid,p2)
            print(x1,y1,x2,y2)
            if x1==x2:
                y1= (y1-1)%5
                y2= (y2-1)%5
            elif y1==y2:
                x1= (x1-1)%5
                x2= (x2-1)%5
            else:
                temp = y1
                y1= y2
                y2= temp
            decryptedtext+=grid[x1][y1]+grid[x2][y2]

    elif (method == 'hill'):
        cyphertext = cyphertext.replace(" ","")
        cyphertext = cyphertext.lower()
    # f) Hill Cipher
        #membuat matrix key
        if m=='':
            #use default m=3 if empty
            m=3
        m=int(m)
        temp_key=''
        if (len_key<m**2):
            for i in range(0,m**2):
                temp_key+= key[i%len_key]
        if (len_key>=m**2):
            temp_key = key[:m**2]
        hill_key = []
        for i in range(0,len(temp_key),m):
            temp_arr=[]
            for j in range(m):
```

```
                k_char = temp_key[i+j]
                k_val = alphabets.index(k_char)
                temp_arr.append(k_val)
            hill_key.append(temp_arr)

        hill_key = np.array(hill_key)
        determinant = np.linalg.det(hill_key)
        det_mod_inv = pow(int(determinant), -1, len(alphabets))
        # hill_key_inv = np.linalg.inv(hill_key)
        # hill_key_inv = (det_mod_inv * hill_key_inv) % 26
        hill_key_inv = det_mod_inv * np.round(determinant *
np.linalg.inv(hill_key)).astype(int) % len(alphabets)

        #perkalian matrix hill cypher
        for i in range(0,len(cyphertext),m):
            c_arr = []
            for j in range (m):
                c_val= alphabets.index(cyphertext[i+j])
                c_arr.append(c_val)
            for row in range(m):
                c_val=0
                for col in range(m):
                    c_val += hill_key_inv[row][col] *c_arr[col]
                decryptedtext+= alphabets[int(c_val)%26]
    return decryptedtext
```

2. File cryptography.html

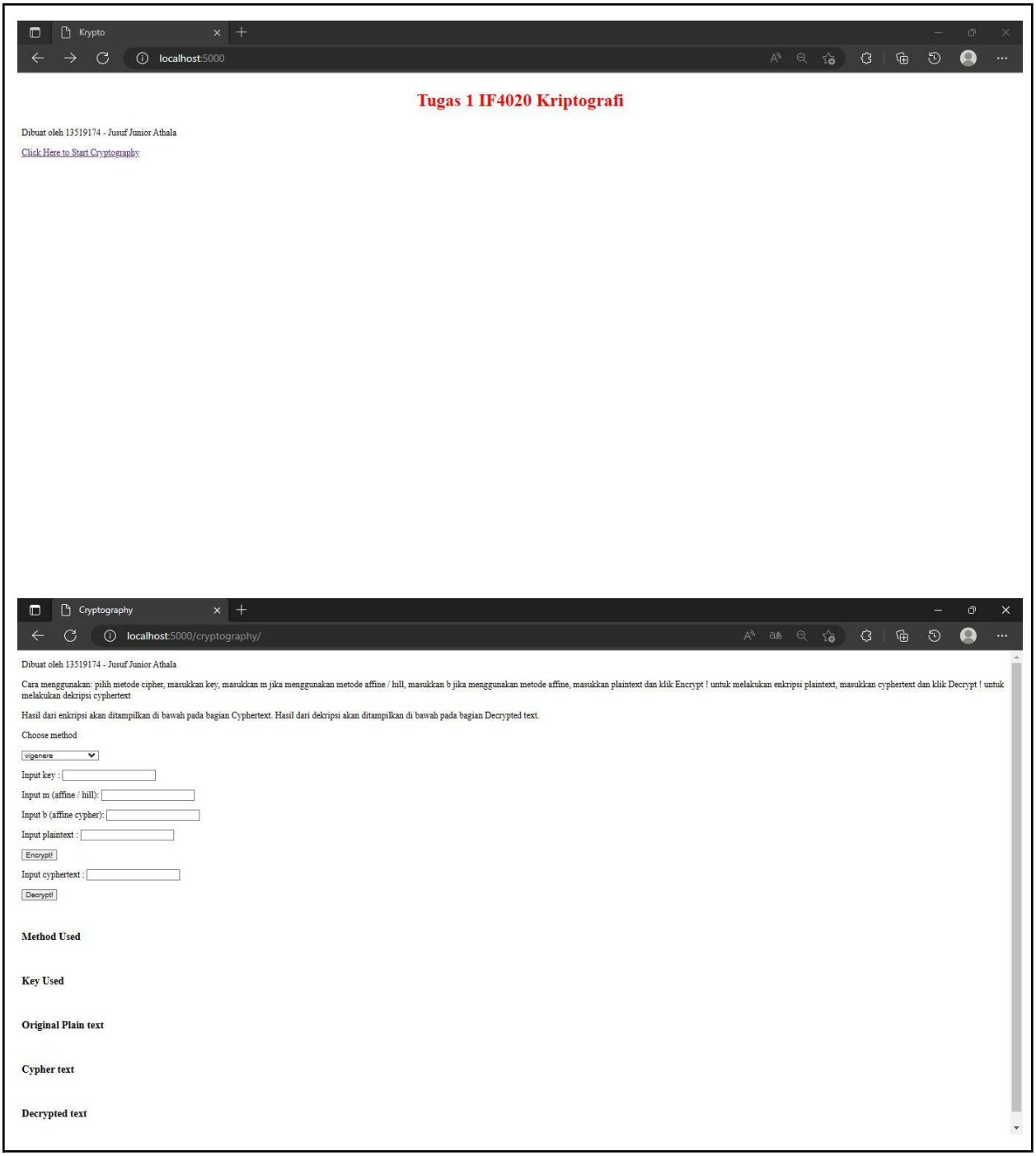
   File ini adalah file html untuk aplikasi. File ini berguna sebagai user interface aplikasi.

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <title>Cryptography</title>
</head>
<body>
    <p> Dibuat oleh 13519174 - Jusuf Junior Athala</p>
    <p> Cara menggunakan: pilih metode cipher, masukkan key,
      masukkan m jika menggunakan metode affine / hill,
      masukkan b jika menggunakan metode affine,
      masukkan plaintext dan klik Encrypt ! untuk melakukan enkripsi plaintext,
      masukkan cyphertext dan klik Decrypt ! untuk melakukan dekripsi
cyphertext
    </p>
    <p> Hasil dari enkripsi akan ditampilkan di bawah pada bagian Cyphertext.
      Hasil dari dekripsi akan ditampilkan di bawah pada bagian Decrypted text.
    </p>
    <p> Ingat ! Setelah mengklik Encrypt! atau Decrypt!, pilihan metode akan
      reset menjadi default vigenere, jangan lupa untuk mengubah metode
      jika anda ingin melakukan encrypt dan decrypt secara berurutan!</p>


    <form action = "{{ url_for("cryptography")}}" method = "POST">

      <p>Choose method</p>
      <select name="method">
        <option value="vigenere">vigenere</option>
        <option value="auto-vigenere">auto-vigenere</option>
        <option value="extended-vigenere">extended-vigenere</option>
        <option value="affine">affine</option>
        <option value="playfair">playfair</option>
        <option value="hill">hill</option>
      </select>
      <p>Input key : <input type = "text" name = "key" value= "{{key}}"/></p>
      <p>Input m (affine / hill): <input type = "number" name = "m" value=
"{{m}}"/></p>
      <p>Input b (affine cypher): <input type = "number" name = "b" value=
"{{b}}"/></p>
      <p>Input plaintext : <input type = "text" name = "plaintext" value=
```

```
"{{plaintext}}"/></p>
      <p><input name = "submit_button" type = "submit" value = "Encrypt!"
/></p>
      <p>Input cyphertext : <input type = "text" name = "cyphertext" value=
"{{cyphertext}}"/></p>
      <p><input name = "submit_button" type = "submit" value = "Decrypt!"
/></p>



     </form>
     <p name = "error_message">{{error_message}}</p>
     <br>

     <h3>Method Used</h3>
     <p>{{method}}</p>
    </br>

     <h3>Key Used</h3>
     <p>{{key}}</p>
    </br>

     <h3>Original Plain text</h3>
     <p>{{plaintext}}</p>
    </br>

     <h3>Cypher text</h3>
     <p>{{result_cyphertext}}</p>
    </br>

     <h3>Decrypted text</h3>
     <p>{{result_decryptedtext}}</p>
    </br>


</body>
</html>
```

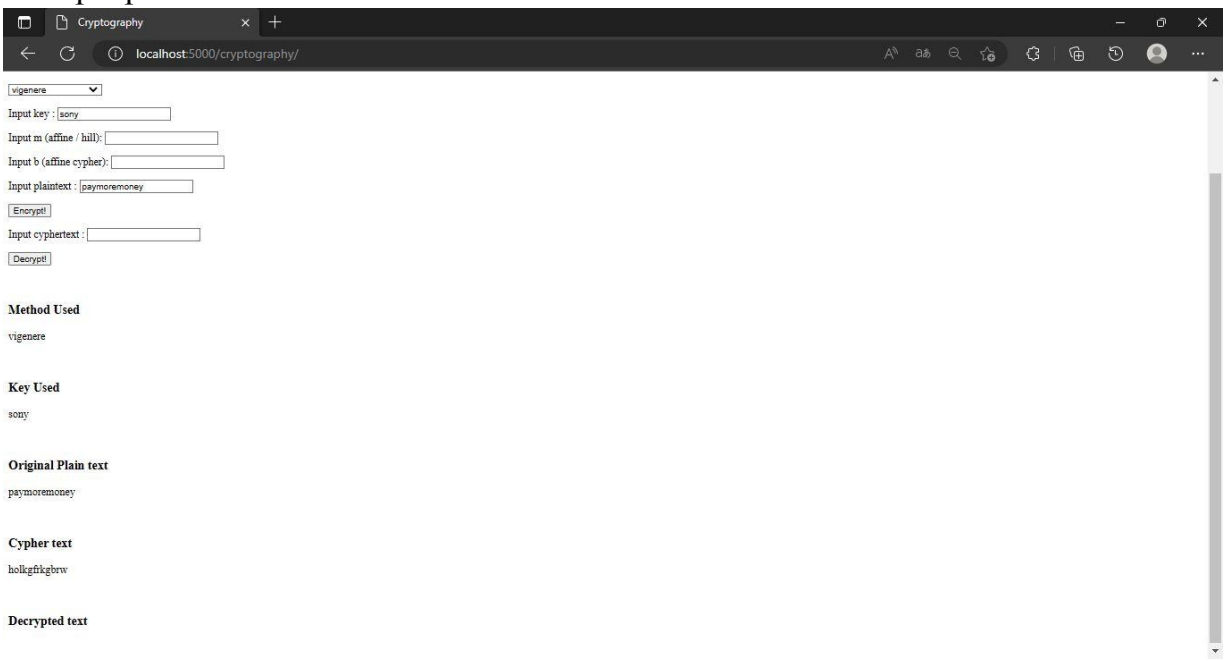
B. Screenshot tampilan antarmuka



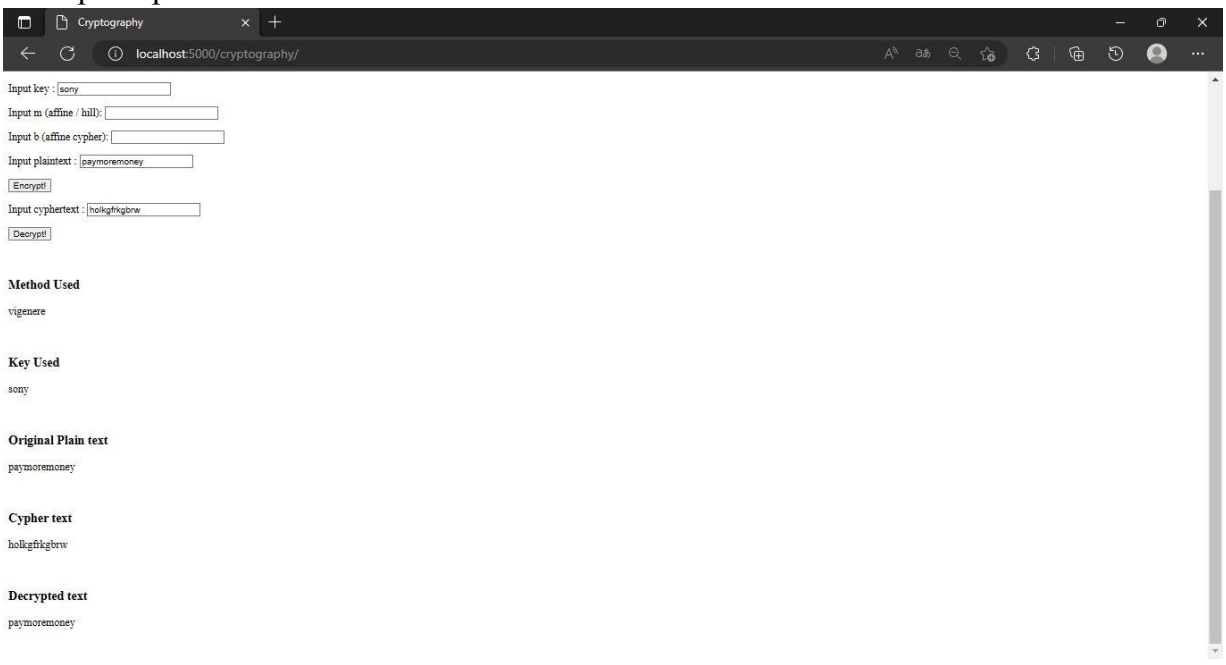
C. Contoh plainteks dan cipherteks

1. Cipher vigenere. Key = sony , plaintext = paymoremoney

Enkripsi plainteks :

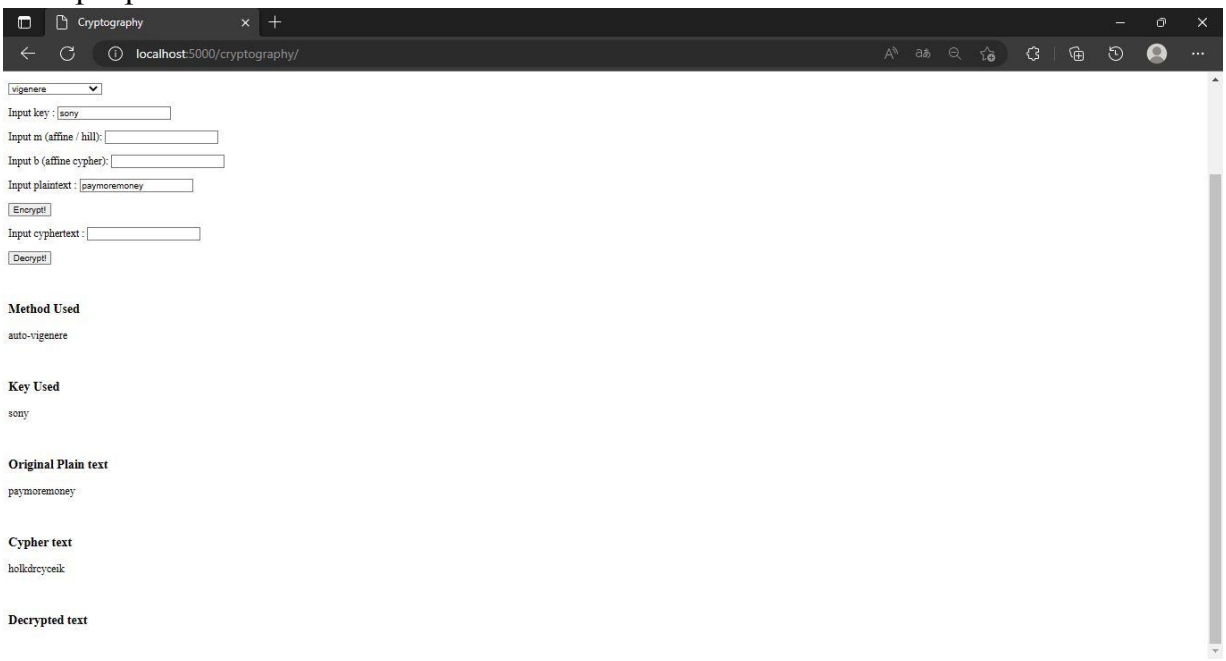



Dekripsi cipherteks

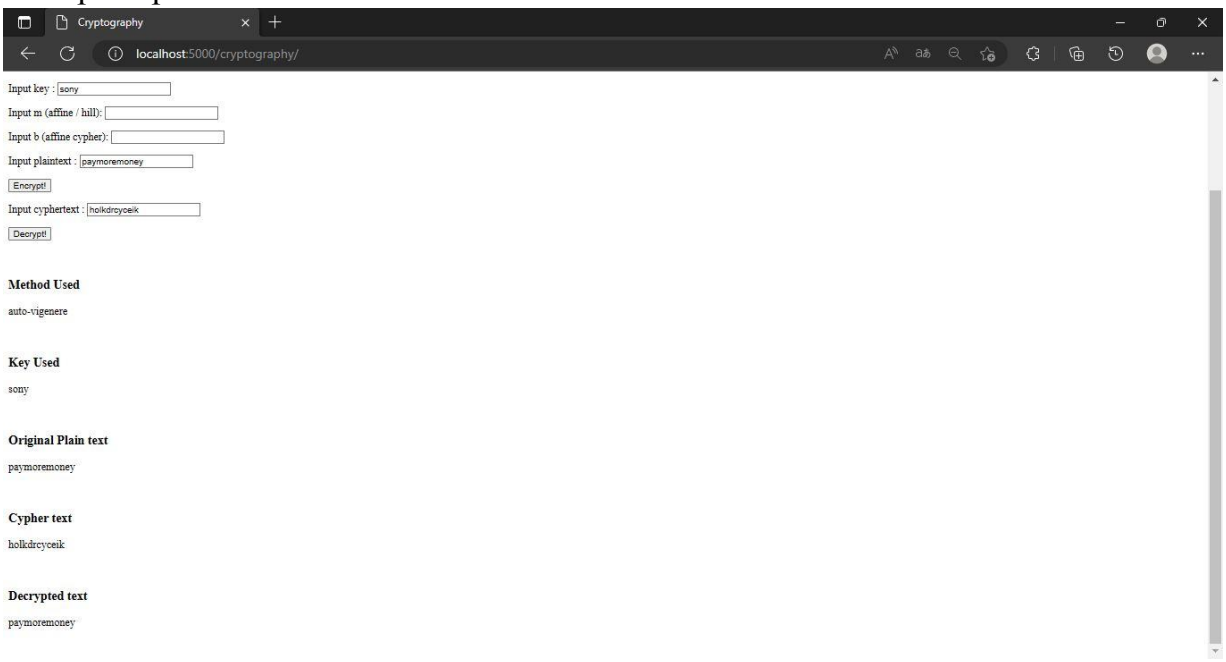



2. Cipher AutoVigenere. Key = sony , plaintext = paymoremoney

Enkripsi plainteks :

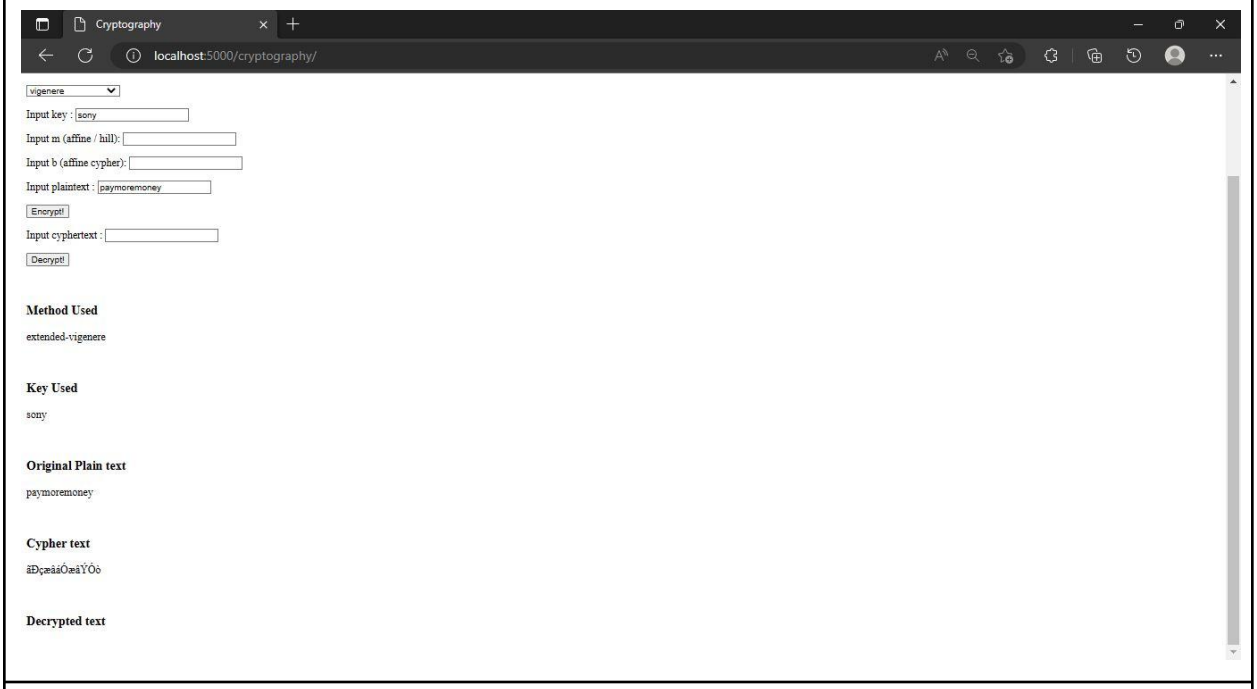



Dekripsi cipherteks

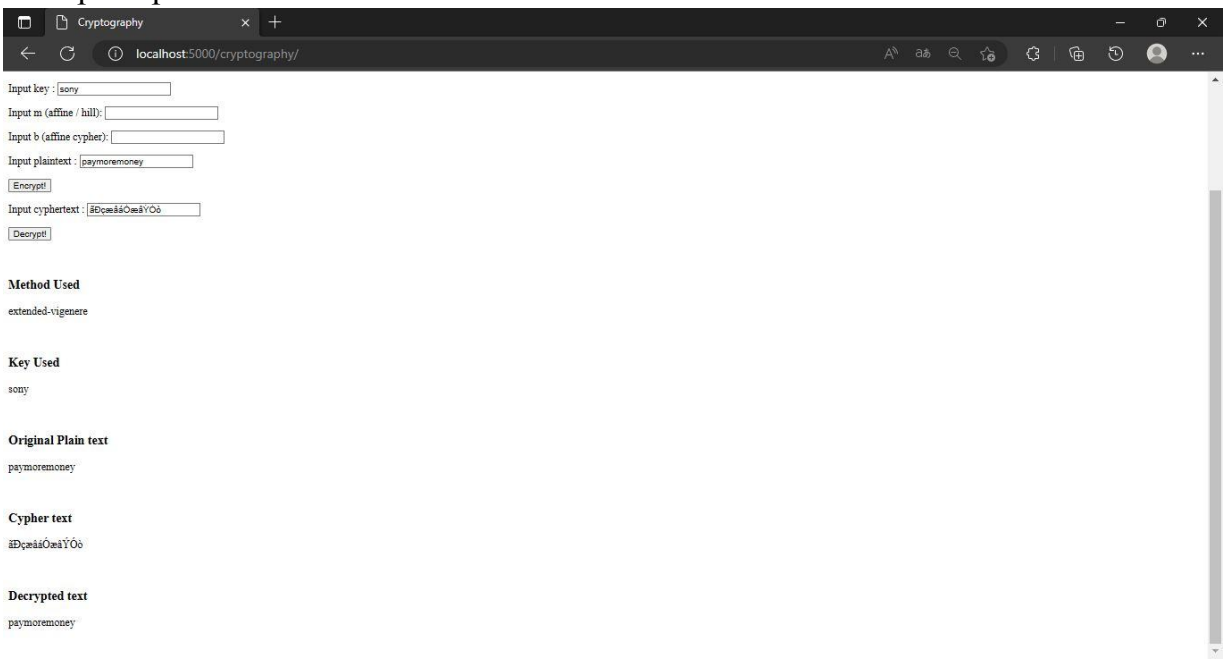



3. Cipher ExtendedVigenere. Key = sony , plaintext = paymoremoney

Enkripsi plainteks :

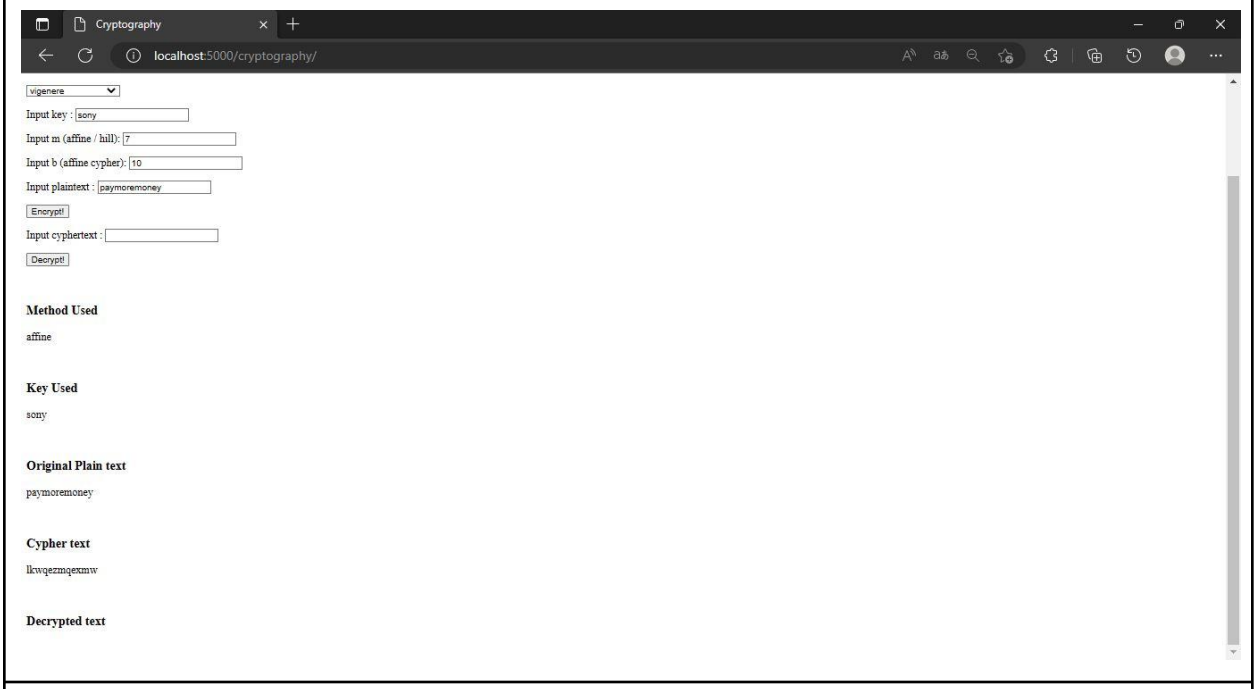



Dekripsi cipherteks

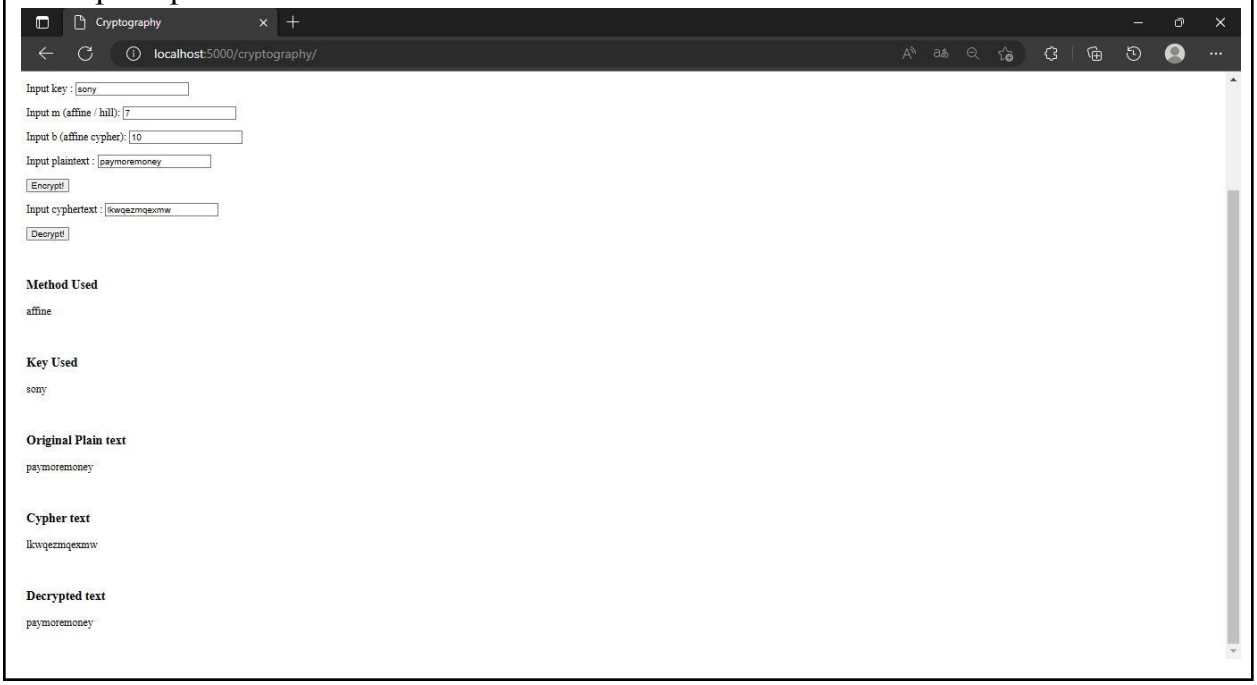



4. Cipher Affine. m=7 , b=10 , plaintext = paymoremoney

Enkripsi plainteks :

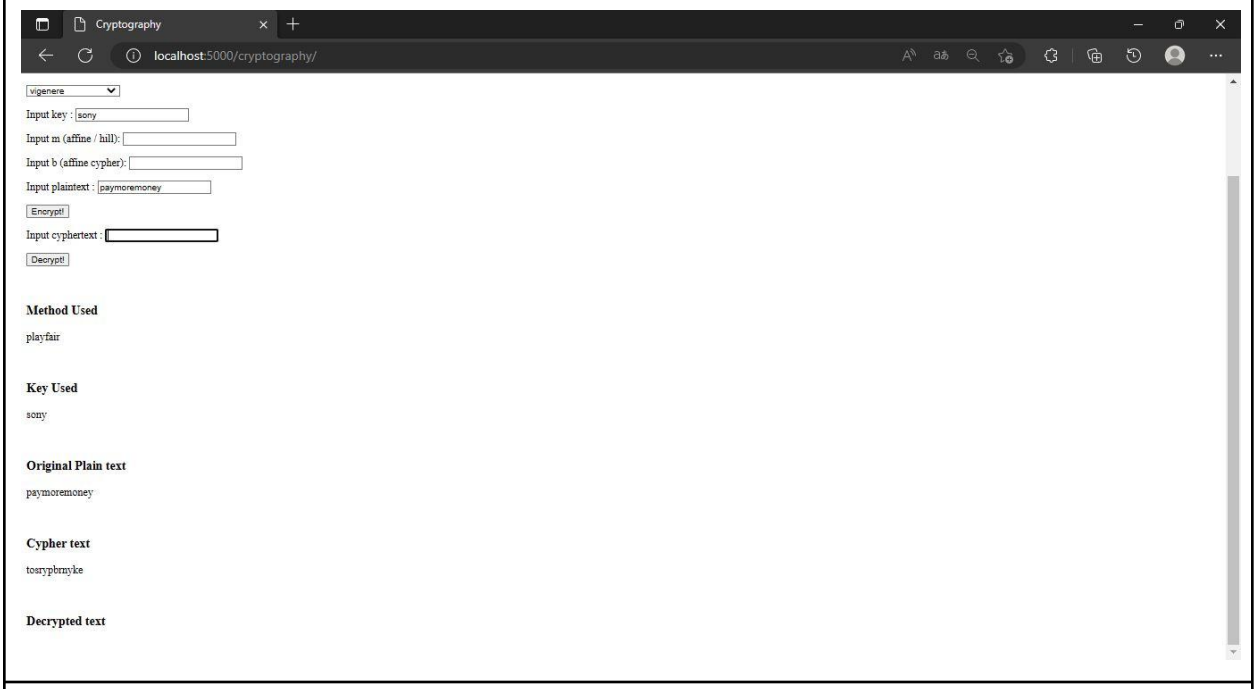



Dekripsi cipherteks

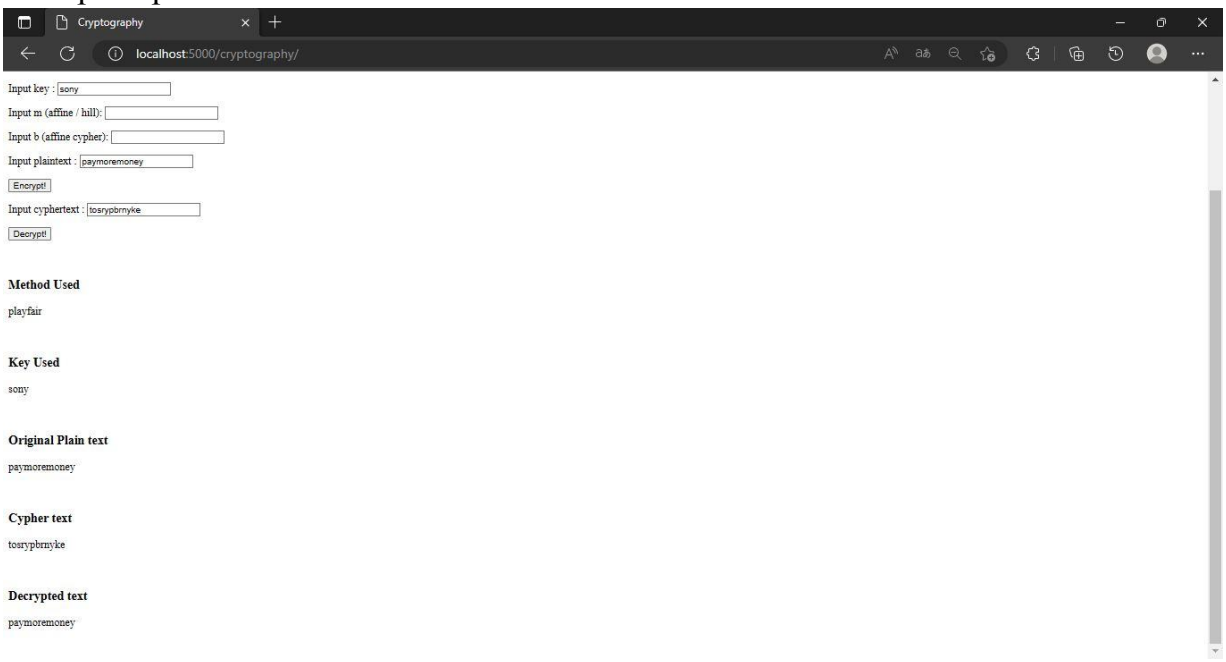



5. Cipher Playfair. Key = sony , plaintext = paymoremoney

Enkripsi plainteks :

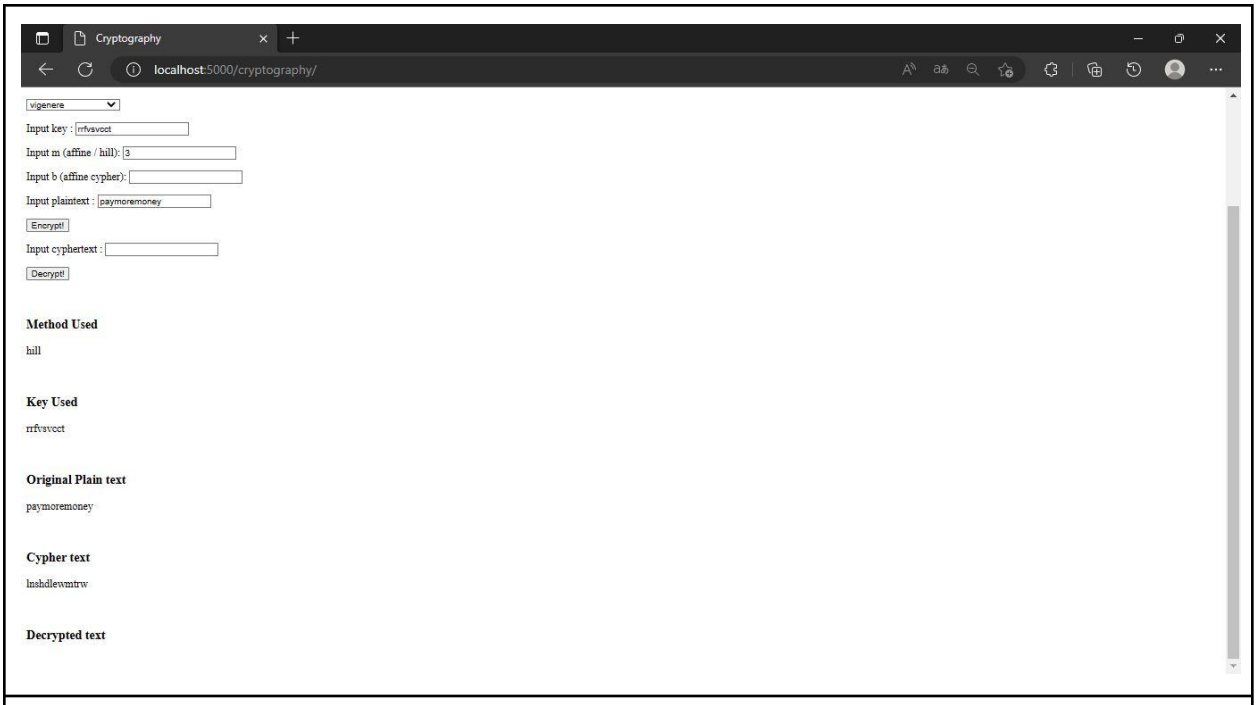



Dekripsi cipherteks

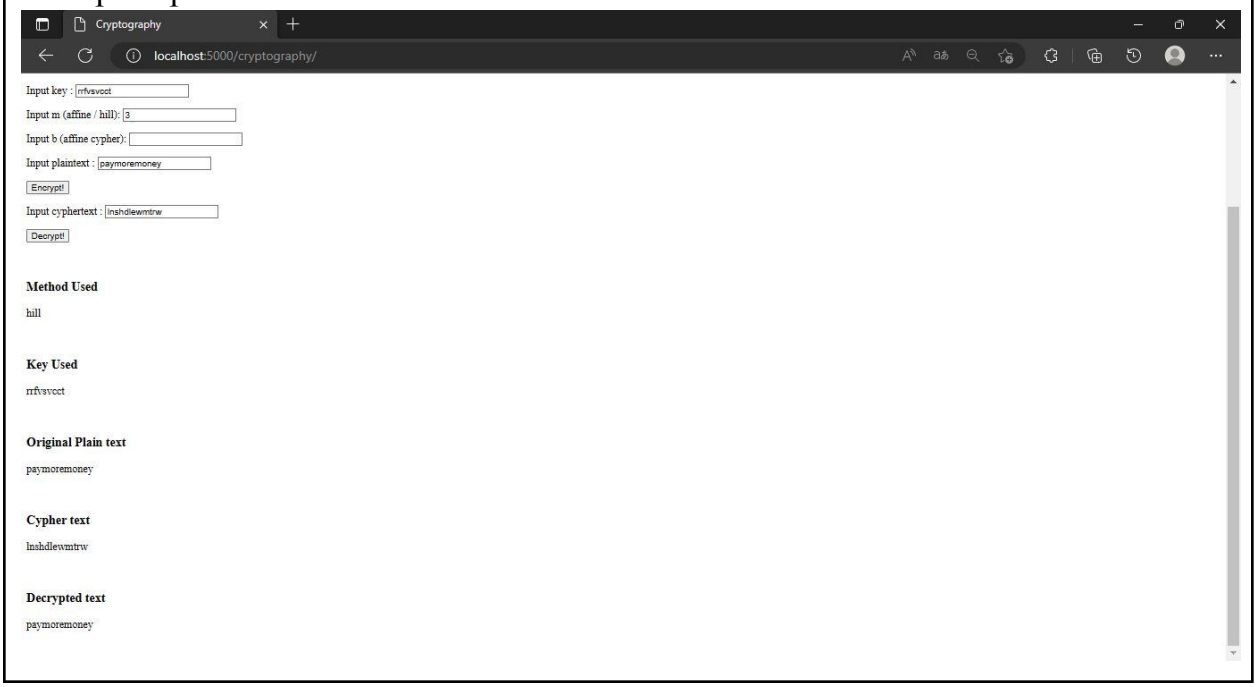



6. Cipher Hill. Key = rrfvsvcct , m =3 , plaintext = paymoremoney

Enkripsi plainteks :




Dekripsi cipherteks




D. Link github program

https://github.com/jusufjathala/if4020-kripto-tugas-1

E. Tabel keberhasilan

Bagian A

| No | Spek | Berhasil (V) | Kurang berhasil (V) | Keterangan |
|---|---|---|---|---|
| 1 | Vigenere standard | | V | hanya bisa enkripsi dan dekripsi pesan diketik langsung tidak bisa untuk file |
| 2 | Auto-Key Vigenere Cipher | | V | hanya bisa enkripsi dan dekripsi pesan diketik langsung tidak bisa untuk file |
| 3 | Extended Vigenere Cipher | | V | hanya bisa enkripsi dan dekripsi pesan diketik langsung tidak bisa untuk file |
| 4 | Affine Cipher | | V | hanya bisa enkripsi dan dekripsi pesan diketik langsung tidak bisa untuk file |
| 5 | Playfair cipher | | V | hanya bisa enkripsi dan dekripsi pesan diketik langsung tidak bisa untuk file |

| | | | | |
|---|---|---|---|---|
| 6 | Hill Cipher | | V | hanya bisa enkripsi dan dekripsi pesan diketik langsung tidak bisa untuk file |
| 7 | Bonus: Enigma cipher | | | Tidak dikerjakan |

Bagian B

| No | Spek | Berhasil (V) | Kurang berhasil (V) | Keterangan |
|---|---|---|---|---|
| 1 | Kriptanalisis Cipher Abjad-Tunggal | | | Tidak dikerjakan |
| 2 | Metode Kasiski | | | Tidak dikerjakan |
| 3 | Kriptanalisis Playfair Cipher | | | Tidak dikerjakan |
| 4 | Kriptanalisis Hill Cipher | | | Tidak dikerjakan |