Acknowledgement

I would like to thank my teacher in charge, Mrs.Vanaja Sambrani for her endless guidance and support towards our project. I would also like to thank our principal, Mrs. Manila Carvalho for providing us with all the resources required to complete this project. I would also like to thank my parents and fellow peers for their encouragement towards this project

Table of Contents

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No** | **Content** | | **Page** |
| 1. | | Project Title | 3 |
| 2. | | Introduction, objective and scope | 3 |
| 3. | | Category | 3 |
| 4. | | Description | 4 |
| 5. | | Future scope and further enhancement | 4 |
| 6. | | Conclusion | 5 |
| 7. | | Project code | 6 |
| 8. | | Project outputs | 16 |
| 9. | | Bibliography | 18 |

Project Details

# Project title

Encryption and Decryption of Data using Matrices

# Introduction

This project involves encoding and decoding the data stored in a given file which is entered by the user into the application. The user can obtain an encoded version of the data file entered. This encoded result is a set of characters which replace the original characters. These set of characters are obtained from a matrix chosen by a unique key entered by the user, which also allots the specific matrix for the given key in order to replace the original data with characters in this specific matrix. The application allows us to obtain the original file from a given file with encoded data (decrypt the data), if the same key is entered into the application, which accesses that specific matrix from which the given data has been encrypted into.

# Objective and Scope

This application allows one to encrypt and decrypt data in order to enhance security of data, which limits the data in the original file to just the individuals who know the unique key to access the specific matrix.

If a given file is encrypted into a specific set of replaced characters, the original set of characters can be obtained only when it’s mapped into the right matrix, via the unique key of that matrix which will be entered by the user.

This prevents data to be accessible by anyone who can retrieve and view the information from a given file, and restricts the availability of the data to selected authorities which can be hand picked according to the user’s convenience. This accessibility is not only barred to anyone who uses the source computer in which the data was made and stored first, followed by encryption, but to any other computer where the encoded file is stored in.

# Category

Python as backend and Tkinter for user interface

# Description

The process of the whole application is mentioned in brief in the following lines.

1. The file with the confidential data is created, and named (as a ***‘.txt‘*** extension file)
2. The user runs the application
3. The **.txt** file with the data to be encrypted is attached to the given application
4. The user enters **‘e’** for encryption and **‘d’** for decryption, after entering the unique key
5. If the user enters **‘e’**, the data in the **.txt** file is encrypted and the output is shown, or if the user enters ‘d’, the encoded data in the **.txt** file is decrypted and the output shown is the original data in the file with the original set of characters to convey the original message
6. When encryption takes place, the unique key maps the given data to a specific matrix which replaces the original set of characters with the characters present in the given matrix in a distinct order, and the result is displayed.
7. When decryption takes place, the unique key maps to the same matrix, and brings out the original set of characters from the replaced one, as they r taken from the same distinct order from the same matrix. And the decrypted result i.e., the original message/information is displayed
8. The save result button saves the given result for the user.

# Future scope and Further Enhancement

Future scope

This application can be used in numerous instances where data must not be revealed to more than a desired number of people. It can be used in keeping information confidential in IT firms, business embassies, schools, colleges etc. In IT firms and businesses, one can use this application to share confidential files only to certain employees, and even if it’s misplaced and reaches hands of any other personnel, it wouldn’t be accessible to them without the unique key and the application. It can even be used by the government to communicate confidential information with other departments and governments. It can be used by schools to share/transport Question papers and other confidential data from one branch/city/school to another, via encryption at one end and decryption at the other. The benefit of having a 9-digit unique key augments the security as its not so unproblematic to predict or hack into. All of this ensures a 2-level safety to the data shared over the internet or any other platform and to who this data reaches.

Further enhancement

A better user-interface can be implied, with more options for the user to utilize, and increase the security of the unique key for the matrix to make it even more safe and secured from data theft.

# Conclusion

This encryption application can be used to keep data secured and safe in a computer system, as well as when it needs to be transferred to one or more other computers. Specific data reaches the hands of chosen authorities, only after being given the unique key to access the original data.

Python Code

**import numpy**

**from tkinter import \***

**from tkinter import filedialog, messagebox**

**import time**

**root = Tk()**

**root.geometry("1200x6000")**

**root.title("Message Encryption and Decryption")**

**Tops = Frame(root, width=1600, relief=SUNKEN)**

**Tops.pack(side=TOP)**

**f1 = Frame(root, width=800, height=700,**

**relief=SUNKEN)**

**f1.pack(side=LEFT)**

**localtime = time.asctime(time.localtime(time.time()))**

**lblInfo = Label(Tops, font=('TimesNewRoman', 50, 'bold'),**

**text="ENCODER-DECODER",**

**fg="Black", bd=10, anchor='w')**

**lblInfo.grid(row=0, column=0)**

**lblInfo = Label(Tops, font=('arial', 20, 'bold'),**

**text=localtime, fg="Steel Blue",**

**bd=10, anchor='w')**

**lblInfo.grid(row=1, column=0)**

**rand = StringVar()**

**key = StringVar()**

**mode = StringVar()**

**Result = StringVar()**

**def qExit():**

**root.destroy()**

**def Reset():**

**rand.set("")**

**key.set("")**

**mode.set("")**

**Result.set("")**

**# file explorer window**

**Msg = []**

**def browseFiles():**

**filename = filedialog.askopenfilename(initialdir="/",**

**title="Select a File",**

**filetypes=(("Text files",**

**"\*.txt\*"),**

**("all files",**

**"\*.\*")))**

**# open the file**

**file = open(filename, 'r')**

**# read the file**

**x = (file.read())**

**# close the file**

**file.close()**

**global Msg**

**Msg = list(x)**

**# Explore button**

**button\_explore = Button(f1, padx=16, pady=8, bd=16, fg="black",**

**font=('arial', 16, 'bold'), width=10,**

**text="Browse Files",**

**command=browseFiles).grid(row=1, column=1)**

**lblMsg = Label(f1, font=('arial', 16, 'bold'),**

**text="FILE", bd=16, anchor="w")**

**lblMsg.grid(row=1, column=0)**

**lblkey = Label(f1, font=('arial', 16, 'bold'),**

**text="KEY", bd=16, anchor="w")**

**lblkey.grid(row=2, column=0)**

**txtkey = Entry(f1, font=('arial', 16, 'bold'),**

**textvariable=key, bd=10, insertwidth=4,**

**bg="powder blue", justify='right')**

**txtkey.grid(row=2, column=1)**

**lblmode = Label(f1, font=('arial', 16, 'bold'),**

**text="MODE(e for encrypt, d for decrypt)",**

**bd=16, anchor="w")**

**lblmode.grid(row=3, column=0)**

**txtmode = Entry(f1, font=('arial', 16, 'bold'),**

**textvariable=mode, bd=10, insertwidth=4,**

**bg="powder blue", justify='right')**

**txtmode.grid(row=3, column=1)**

**lblService = Label(f1, font=('arial', 16, 'bold'),**

**text="The Result-", bd=16, anchor="w")**

**lblService.grid(row=2, column=2)**

**txtService = Entry(f1, font=('arial', 16, 'bold'),**

**textvariable=Result, bd=10, insertwidth=4,**

**bg="powder blue", justify='right')**

**txtService.grid(row=2, column=3)**

**def matrix(x, c):**

**y = len(x) // 3**

**z = 0**

**for b in range(0, y):**

**c.append(x[z:z + 3])**

**z = z + 3**

**return c**

**R = []**

**Q = []**

**mtrx2 = []**

**k = key.get()**

**m = mode.get()**

**def encode(key, clear):**

**print("Message= ", (Msg))**

**clear = Msg**

**global k**

**global m**

**j = list(clear)**

**enc = []**

**for i in j:**

**enc.append(ord(i))**

**enc.extend([0] \* (3 - (len(enc) % 3)))**

**mtrx = []**

**matrix(x=enc, c=mtrx)**

**msg2 = []**

**for z in str(key):**

**msg2.append(int(z))**

**global mtrx2**

**matrix(x=msg2, c=mtrx2)**

**rslt = []**

**result = numpy.dot(mtrx, mtrx2)**

**for v in result:**

**rslt.append(v)**

**for i in range(0, len(rslt)):**

**r = rslt[i] // 256**

**global Q**

**Q.append(r)**

**i += 1**

**for i in range(0, len(rslt)):**

**r = rslt[i] % 256**

**global R**

**R.append(r)**

**i += 1**

**enc = ""**

**for i in range(0, len(R)):**

**for j in range(0, 3):**

**enc += chr(R[i][j])**

**return enc**

**clear = Msg**

**m = mode.get()**

**k = key.get**

**def decode(key, enc):**

**print("Message= ", (Msg))**

**global clear**

**global m**

**global k**

**t = []**

**global R**

**for i in range(0, len(R)):**

**t.extend((256 \* Q[i]) + R[i])**

**mtrx3 = []**

**matrix(x=t, c=mtrx3)**

**loo = numpy.array(mtrx3, dtype=int)**

**ferrari = []**

**global mtrx2**

**honda = numpy.dot(loo, numpy.linalg.inv(mtrx2))**

**for g in honda:**

**ferrari.append(g)**

**mclaren = []**

**for i in range(0, len(ferrari)):**

**for j in range(0, 3):**

**mclaren.append(round(ferrari[i][j]))**

**dec = ""**

**for k in range(0, len(mclaren)):**

**dec += chr(mclaren[k])**

**return dec**

**def Ref():**

**clear = Msg**

**k = key.get()**

**m = mode.get()**

**if (m == 'e'):**

**result = encode(k, clear)**

**Result.set(result)**

**# Result.set(encode(k, clear))**

**else:**

**final = decode(key, clear)**

**Result.set(final)**

**def saveResultToFile():**

**result\_text = Result.get()**

**if result\_text:**

**filename = filedialog.asksaveasfilename(defaultextension=".txt", filetypes=[("Text files", "\*.txt\*"), ("All files", "\*.\*")])**

**with open(filename, 'w', encoding='utf-8') as file: # Specify the encoding as 'utf-8'**

**file.write(result\_text)**

**messagebox.showinfo("Saved", "Result saved to {}".format(filename))**

**else:**

**messagebox.showwarning("No Result", "No result to save. Please generate a result first.")**

**btnSave = Button(f1, padx=16, pady=8, bd=16, fg="black",**

**font=('arial', 16, 'bold'), width=10,**

**text="Save Result", bg="yellow",**

**command=saveResultToFile).grid(row=7, column=4)**

**btnTotal = Button(f1, padx=16, pady=8, bd=16, fg="black",**

**font=('arial', 16, 'bold'), width=10,**

**text="Show Message", bg="powder blue",**

**command=Ref).grid(row=7, column=1)**

**btnReset = Button(f1, padx=16, pady=8, bd=16,**

**fg="black", font=('arial', 16, 'bold'),**

**width=10, text="Reset", bg="green",**

**command=Reset).grid(row=7, column=2)**

**btnExit = Button(f1, padx=16, pady=8, bd=16,**

**fg="black", font=('arial', 16, 'bold'),**

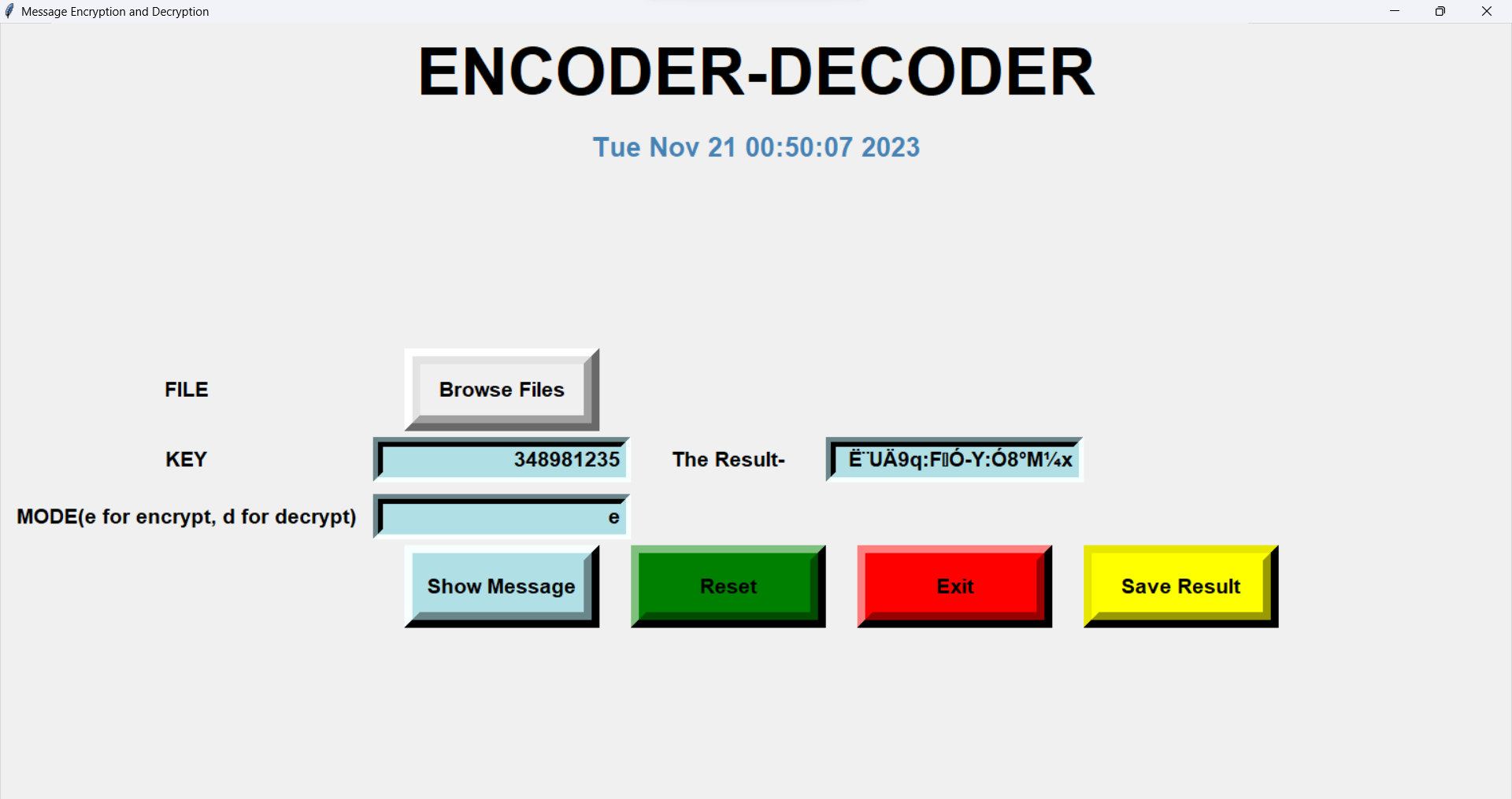
**width=10, text="Exit", bg="red",**

**command=qExit).grid(row=7, column=3)**

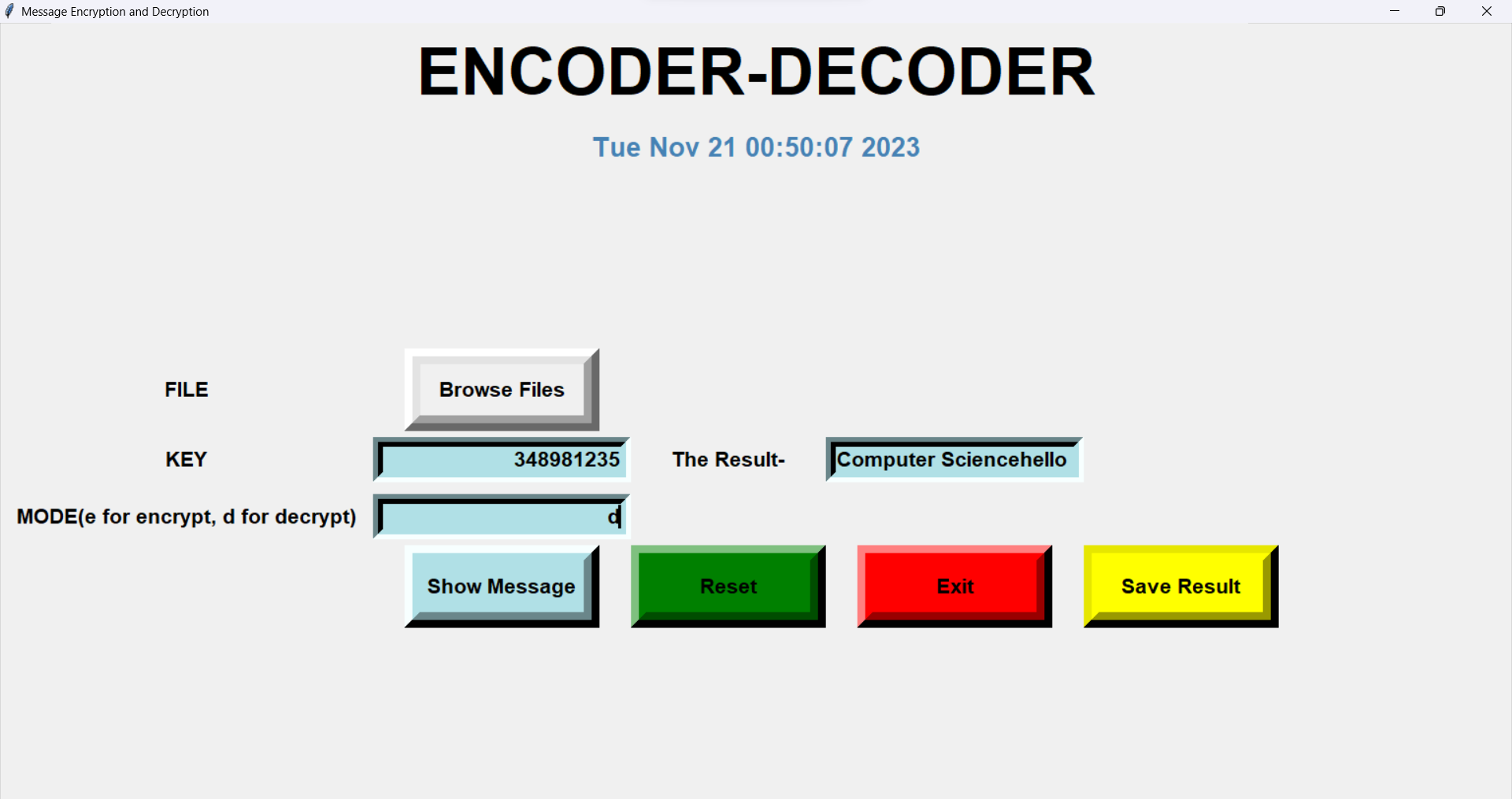
**root.mainloop()**

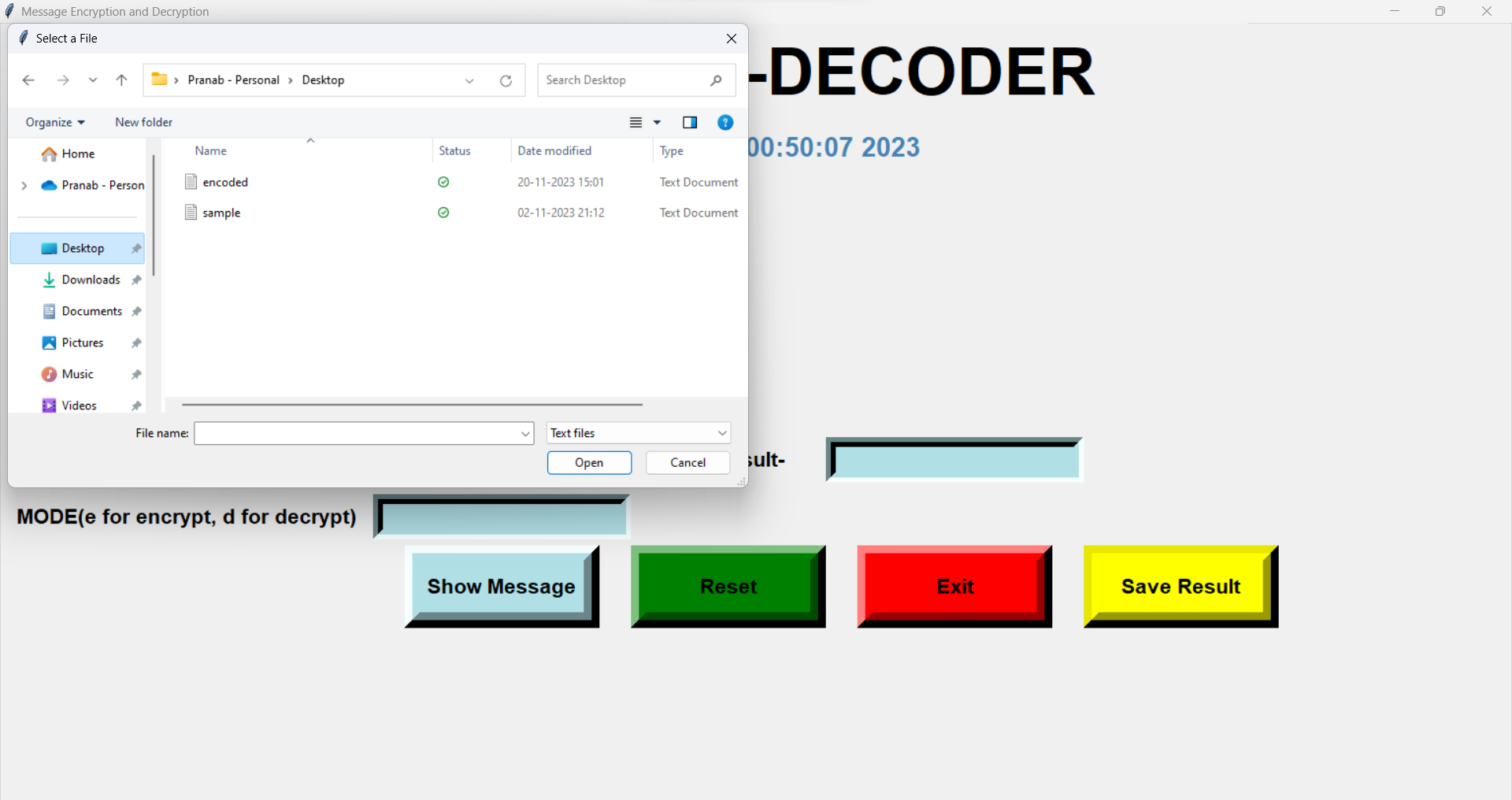
**Code Outputs**

Encryption



Decryption



Browsing

Bibliograpy

* COMPUTER SCIENCE WITH PYTHON - BY SUMITA ARORA
* www.datacamp.com