A

PROJECT

ON

"ArchSafe"

Submitted to

Shiksha Mandal's

G. S. COLLEGE OF COMMERCE & ECONOMICS, NAGPUR

(AUTONOMOUS)

In the Partial Fulfillment of

B.Com. (Computer Application) Final Year

Submitted by

Chandrakanha Dharmik

Nachiket Patil

Under the Guidance of

Pravin J. Yadao

Shiksha Mandal's

G. S. COLLEGE OF COMMERCE & ECONOMICS, NAGPUR

(AUTONOMOUS)

2021-2022

Shiksha Mandal's

G. S. COLLEGE OF COMMERCE & ECONOMICS, NAGPUR (AUTONOMOUS)

CERTIFICATE

(2021 - 2022)

This is to certify that Mr. Chandrakanha Dharmik and Nachiket Patil have completed their project on the topic of ArchSafe prescribed by G. S. College of Commerce & Economics, Nagpur (Autonomous) for B.Com. (Computer Application) – Semester-VI.

Date:

Place: Nagpur

Pravin J. Yadao

Project Guide

External Examiner **Internal Examiner**

ACKNOWLEDGEMENT

We take this opportunity to express our deep gratitude and whole hearted thanks to project guide Prof. Pravin Yadao, Coordinator for his guidance throughout this work. We are very much thankful to him for his constant encouragement, support and kindness.

We are also grateful to our teachers Prof. Rahul Tiwari, Prof. Sushma Gawande, Prof. Preeti Rangari, Prof. Prajkta Deshpande and Prof. Haresh Naringe for their encouragement, help and support from time to time.

We also wish to express our sincere thanks to Principal Dr. N. Y. Khandait for providing us wide range of opportunities, facilities and inspiration to gather professional knowledge and material without which this project could not have been completed.

Date: Chandrakanha Dharmik

Place: Nagpur

Nachiket Patil

DECLARATION

We **Chandrakanha Dharmik & Nachiket Patil** hereby honestly declare that the work entitled **"ArchSafe"** submitted by us at G. S. College of Commerce & Economics, Nagpur (Autonomous) in partial fulfillment of requirement for the award of B.Com. (Computer Application) degree by Rashtrasant Tukadoji Maharaj, Nagpur University, Nagpur has not been submitted elsewhere for the award of any degree, during the academic session 2021-2022.

The project has been developed and completed by us independently under the supervision of the subject teacher and project guide.

Chandrakanha Dharmik

Nachiket Patil

Date:

Place: Nagpur

Introduction

Introduction

In today's age of digitalization, Data holds the most important key to any organization's functioning and value. Any data that is held by an organization leads to its growth, stability, security, future plans, growth, analysis etc. In the evolving world of data and information transfer, security of the file contents remains to be one of the greatest concerns for companies. Some information can be password protected (emails, logins) while other information being transferred via emails or FTP lacks efficiency if protected by some keyword. This is where file encryption plays a big role and provides security and convenience sought by parties engaged in file transfers.

To overcome this there is a very simple yet vastly efficient way to secure the data and its transference either in the organization or from one branch to another. This method is called, Encryption of data. The term **Encryption** stands for securing of data by converting it into a form that is only accessible to a certain group of people or someone who is actually authorized to view, edit, update that particular data in the organization.

So, what is Encryption? It is a process of converting information into some form of a code to hide its true content. The only way to access the file information then is to *decrypt* it. The process of encryption/decryption is called cryptography.

At the beginning of the encryption process, the sender must decide what cipher will best disguise the meaning of the message and what variable to use as a key to make the encoded message unique. The most widely used types of ciphers fall into two categories: symmetric and asymmetric.

Symmetric ciphers, also referred to as secret key encryption, use a single key. The key is sometimes referred to as a shared secret because the sender or computing system doing the encryption must share the secret key with all entities authorized to

decrypt the message. Symmetric key encryption is usually much faster than asymmetric encryption. The most widely used symmetric key cipher is the Advanced Encryption Standard (AES), which was designed to protect government-classified information.

Asymmetric ciphers, also known as public key encryption, use two different -- but logically linked -- keys. This type of cryptography often uses prime numbers to create keys since it is computationally difficult to factor large prime numbers and reverse-engineer the encryption. The Rivest-Shamir-Adleman (RSA) encryption algorithm is currently the most widely used public key algorithm. With RSA, the public or the private key can be used to encrypt a message; whichever key is not used for encryption becomes the decryption key.

Today, many cryptographic processes use a symmetric algorithm to encrypt data and an asymmetric algorithm to securely exchange the secret key. The primary purpose of encryption is to protect the confidentiality of digital data stored on computer systems or transmitted over the internet or any other computer network. In addition to security, the adoption of encryption is often driven by the need to meet compliance regulations. A number of organizations and standards bodies either recommend or require sensitive data to be encrypted in order to prevent unauthorized third parties or threat actors from accessing the data. For example, the Payment Card Industry Data Security Standard (PCI DSS) requires merchants to encrypt customers' payment card data when it is both stored at rest and transmitted across public networks.

History of encryption

The word encryption comes from the Greek word kryptos, meaning hidden or secret. The use of encryption is nearly as old as the art of communication itself. As early as 1900 B.C., an Egyptian scribe used nonstandard hieroglyphs to hide the meaning of an inscription. In a time when most people couldn't read, simply writing a message was often enough, but encryption schemes soon developed to convert messages into unreadable groups of figures to protect the message's secrecy while it was carried from one place to another. The contents of a message were reordered (transposition) or replaced (substitution) with other characters, symbols, numbers or pictures in order to conceal its meaning.

In 700 B.C., the Spartans wrote sensitive messages on strips of leather wrapped around sticks. When the tape was unwound, the characters became meaningless, but with a stick of exactly the same diameter, the recipient could recreate (decipher) the message. Later, the Romans used what's known as the Caesar Shift Cipher, a monoalphabetic cipher in which each letter is shifted by an agreed number. So, for example, if the agreed number is three, then the message, "Be at the gates at six" would become "eh dw wkh jdwhv dw vla." At first glance, this may look difficult to decipher, but juxtaposing the start of the alphabet until the letters make sense doesn't take long. Also, the vowels and other commonly used letters, like t and s, can be quickly deduced using frequency analysis, and that information, in turn, can be used to decipher the rest of the message.

The Middle Ages saw the emergence of polyalphabetic substitution, which uses multiple substitution alphabets to limit the use of frequency analysis to crack a cipher. This method of encrypting messages remained popular despite many implementations that failed to adequately conceal when the substitution changed --

also known as key progression. Possibly the most famous implementation of a polyalphabetic substitution cipher is the Enigma electromechanical rotor cipher machine used by the Germans during World War II.

It was not until the mid-1970s that encryption took a major leap forward. Until this point, all encryption schemes used the same secret for encrypting and decrypting a message: a symmetric key.

Encryption was almost exclusively used only by governments and large enterprises until the late 1970s when the Diffie-Hellman key exchange and RSA algorithms were first published and the first PCs were introduced.

In 1976, Whitfield Diffie and Martin Hellman's paper, "New Directions in Cryptography," solved one of the fundamental problems of cryptography: how to securely distribute the encryption key to those who need it. This breakthrough was followed shortly afterward by RSA, an implementation of public key cryptography using asymmetric algorithms, which ushered in a new era of encryption. By the mid-1990s, both public key and private key encryption were being routinely deployed in web browsers and servers to protect sensitive data.

Objectives

Objectives

- User friendly application for secure transmission.

Basic aim is to develop a user-friendly application so that user can securely encrypt data or

information with limited knowledge about cryptographic algorithms. Any individual or

organization can rely on this application for confidentiality and authenticity of resources

- Easy implementation of cryptographic function.
- Cryptographic libraries are based on command line tools and are difficult to be used. It requires sequential instruction to be provided manually through windows forms. This application will make the use of crypto function through interface
- Combination of different functionalities

Application reduces the effort of executing commands one after another enabling user to view,

control, and manipulate multiple things simultaneously. Also application executes multiple

tasks to be performed in one step.

- Ensured higher level protection against cyber attacks

As the login credentials are stored locally, it’s easy to maintain a higher level of security as the database itself will be protected by a password not accessible to anyone else other than the creators of the software.

- Easily portable and installable

This application can be run and used by everyone across multiple windows devices. This, even if it is platform dependency, assures that it will be able to run across multiple devices and installable in windows devices without any extra software or plugin to understand the working or the running of the application.

- Security

This application aims at providing a greater security to the user by restricting the user to enter the contact somebody has already registered wherewith. The forget password facility ask the user to enter the security question and answer which he/she has been entered at the registration wrong question and answer doesn't allow his/her to proceed.

Preliminary System Analysis

Preliminary System Analysis

Preliminary system analysis is a process of collecting factual data, understand the processes involved, identifying problems and recommending feasible suggestions for improving the system functioning. This involves studying the business processor. Gathering operational data, understand the information flow, finding out bottlenecks and evolving solutions for overcoming the weakness of the system so as to achieve the organizational goals. System analysis also includes sub-dividing of complex process involving the entire system, identification of data source and annual processes.

Preliminary Investigation

In this, process the development team visits the customer and studies their system they investigate the need of the possible software automation in the given system by the end of preliminary investigation, the team furnishes a document that holds a different specific recommendation for the candidate system.it also includes personal assignment cost, project schedule, target dates, main task of the preliminary investigation phase are:

1) Investigation the present system and identify the function to be perform.

2) Identify the objectives of new system in the general, an information system benefits a business increasing efficiency, improving effectiveness, or providing a competitive advantage.

3) Identify problems and suggests a few solutions.

4) It is organized combination of different components.

5) They are independent and inter-related.

Present System in Use

Currently many softwares provide the text encryption facility which provides a basic Caesar's Cipher format of algorithm/key to encrypt data.

Flaws in present system

Mostly they require a very high subscription amount or the ones that are free aren't that worthy and some do not even allow you to edit info on the go as they only accept files and not text directly.

Needs of new system

- **Confidentiality** encodes the message's content without leaking the key
- **Authentication** verifies that the encrypted hasn't been tampered with.
- **Integrity** proves the contents of a message have not been changed since it was encrypted.
- **Nonrepudiation** prevents senders from denying that they encrypted the message as the database is locally stored and it is also inaccessible.

Feasibility Study

Feasibility study is the preliminary study undertaken before the real work of the project starts to ascertain the like hood of the project success. It analyzes the possible solutions to a problem and a recommendation on the best solutions to use. It involves the evaluation that how the solution will fit into the corporation. A Feasibility study is defined as an evolution or analysis of the potential impacts of a proposed project or system. A feasibility study is conducted to assist decision makers in determining whether or not to implement a particular project or system.

On the basis of result of the initial study, feasibility study takes place. The feasibility study is basically the proposed system in the lights of its workability, meeting user's requirements, and effective use of resources and of course, cost effectiveness. The main goal of feasibility study is not to solve the problem but to achieve this scope. In the process of feasibility study, the cost and benefits are estimated with the greater accuracy. It evaluates the benefits of the new system. The feasibility study will contain the extensive data related to financial and operational impact and will include advantage and disadvantages of both current situation and plan.

The aim of feasibility study is to see whether it is possible to develop a reasonable cost. At the end of feasibility study a decision is taken whether or proceed or not.

Feasibility study is to determine various solution of the problem and then picking up one of the best solutions. It is the measure of how beneficial the development of information system will be to an organization. The study also shows the sensitivity of business to change in the basic assumption.

Economic Feasibility

For any system if the expected benefits equal or exceed the expected costs, the system can be judged to be economically feasible. In economic feasibility, cost benefit analysis is done in which expected costs and benefits are evaluated.

Economic analysis is used for evaluation of the effectiveness of the proposed system.

In this type of feasibility study, the most important is cost and benefit analysis. As the name suggests, it is as analysis of the costs to be incurred in the system and benefits derivable out of the system.

Technical Feasibility

In technical feasibility the following issues are taken into consideration.

- Whether the required technology is available or not.
- Whether the required resources are available like manpower, programmers, testers and debuggers, software and hardware.

Social Feasibility

The affect that a proposed system may have on the social system in the project environment is addressed in the social feasibility. It may happen that particular category of employees may be short or not available as a result of ambient structure. The influence on the social status of the participants by the project should be evaluated on order to guarantee compatibility. It must be identified that the employees in the particular industries may have specific status symbols within the society.

Behavioral Feasibility

It includes how strong the reaction of staff will be towards the development of new system that involves computer's use in their daily work. So resistant to change is identified. It considers human issue. All system development projects introduce change, and people generally resist change. Over resistance from employees may take the form of subrogating the new system (e.g., entering data incorrectly) or deriding the new system to anyone who will listen. Convert resistance typically occurs when employees simply do their jobs using their old methods.

Behavioral feasibility is concerned with assessing the skills and the training needed to use the new is. In some organizations, a proposed system may require mathematical or linguistic skills beyond what the workforce currently processes. In other words, a workforce may simply need to improve their skills. Behavioral feasibility is as much about "can they use it" as it is about "will they use it".

After the feasibility analysis, a "Go/No-Go" decision is reached. The project sponsor and project manager sign off on the decision. If it is a no-go decision, the project is put on the shelf until condition are favorable. Or the project is discarded. Of the decision is "go", then the system development project proceeds.

Project Category

Project Category

In this project, mainly a GUI module of Python (known as tkinter) has been used for frontend and backend. It also requires database connectivity which is fulfilled by using MySQL.

It is a windows application.

What is Python? Executive Summary

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

What is MySQL?

MySQL is a relational database management system (RDBMS) developed by Oracle that is based on structured query language (SQL).

A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or a place to hold the vast amounts of information in a corporate network. In particular, a relational database is a digital store collecting data and organizing it according to the relational model. In this model, tables consist of rows and columns, and relationships between data elements all follow a strict logical structure. An RDBMS is simply the set of software tools used to actually implement, manage, and query such a database.

MySQL is integral to many of the most popular software stacks for building and maintaining everything from customer-facing web applications to powerful, data-driven B2B services. Its open-source nature, stability, and rich feature set, paired with ongoing development and support from Oracle, have meant that internet-critical organizations such as Facebook, Flickr, Twitter, Wikipedia, and YouTube all employ MySQL backends.

Software & Hardware Requirement Specifications

Software & Hardware Requirement Specifications

Every application needs the software in which it has to be executed and a hardware the application is going to perform its function. Some application cannot run on every platform and some applications needs some specific requirement in the software or in hardware to get operated. Let's take an example of the applications which cannot be run on every platform like windows, android, Linux, etc. Applications made in visual basic is only supported for the windows, one cannot access these applications from the mobile phones, etc. So, here are some hardware and software specifications which are mandatory for the application to get operated.

Hardware

Hardware is a term that refers to all the physical parts that make up a computer. The internal hardware devices that make up the computer. Various devices which are essentials to form a hardware is called as components.

Following are the hardware specifications that is required to develop this project is as follows:

Computer components like Monitor, Keyboard, Mouse, CPU, Keyboard.

Minimum 1 GB ram for smooth working of application.

250 GB Hard Disk or More.

CD ROM Drive.

Software

Software can be termed as the group of instruction or command used by the computer to accomplish the given task.

It can be said as a set of instructions or programs instructing a computer to do specific task. Software in general term is used to describe the computer programs.

Following are the software specifications that is required to develop this project is as follows:

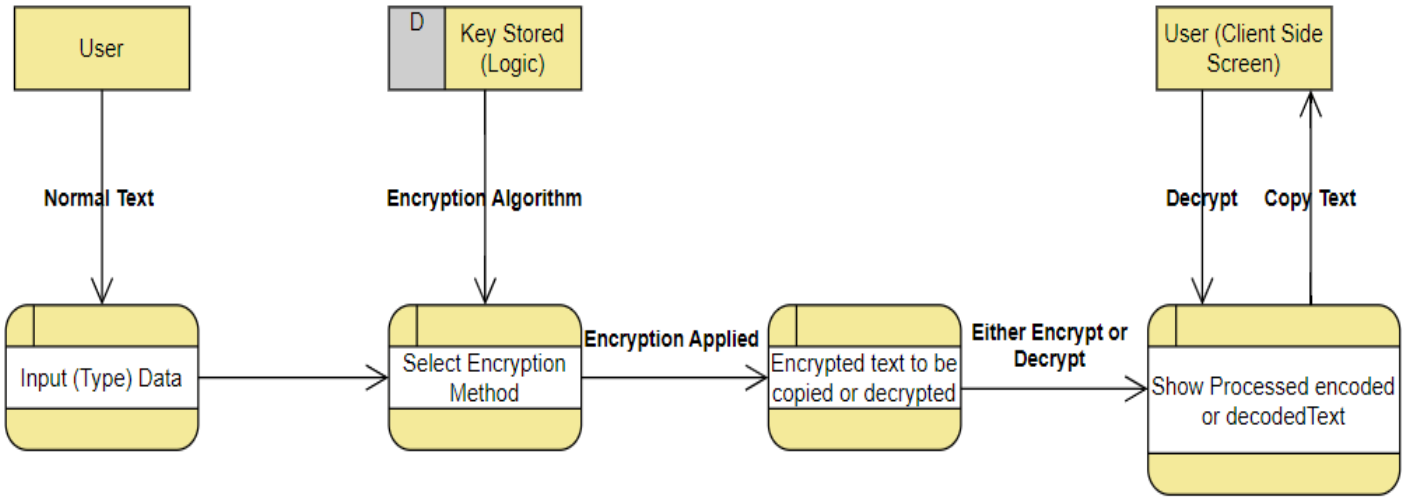
Operating System: Microsoft Windows XP or above versions.

Language Used (Front End): Python

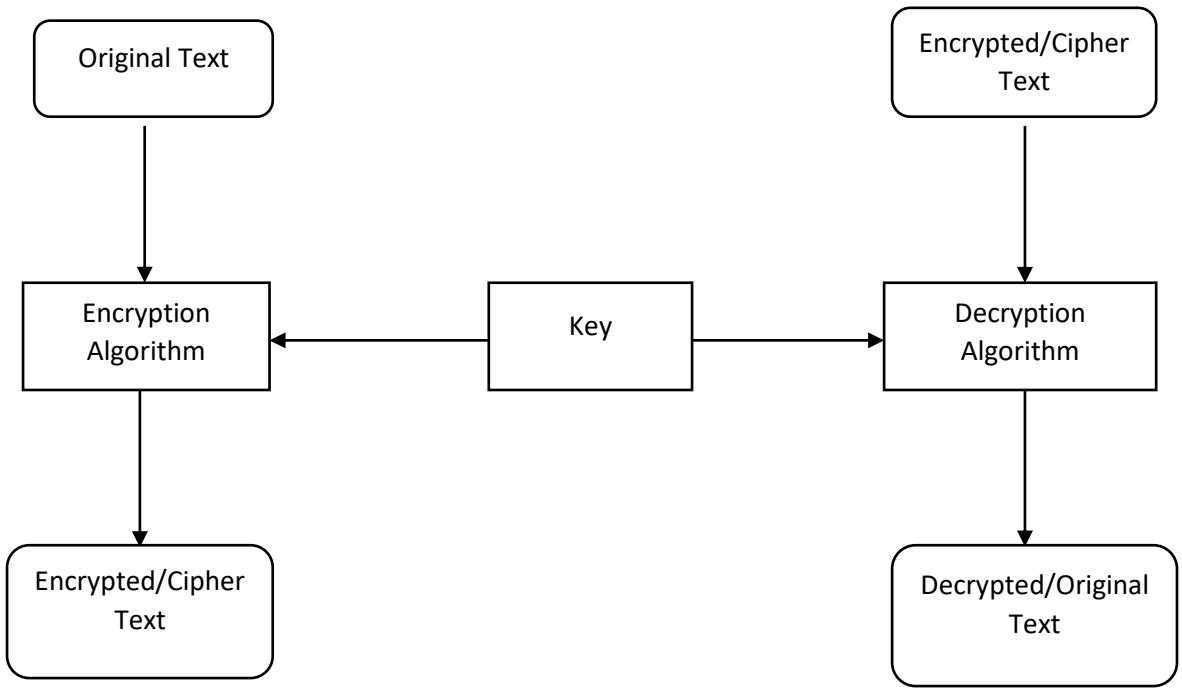
Database Used (Back End): MySQL

Detailed System Analysis

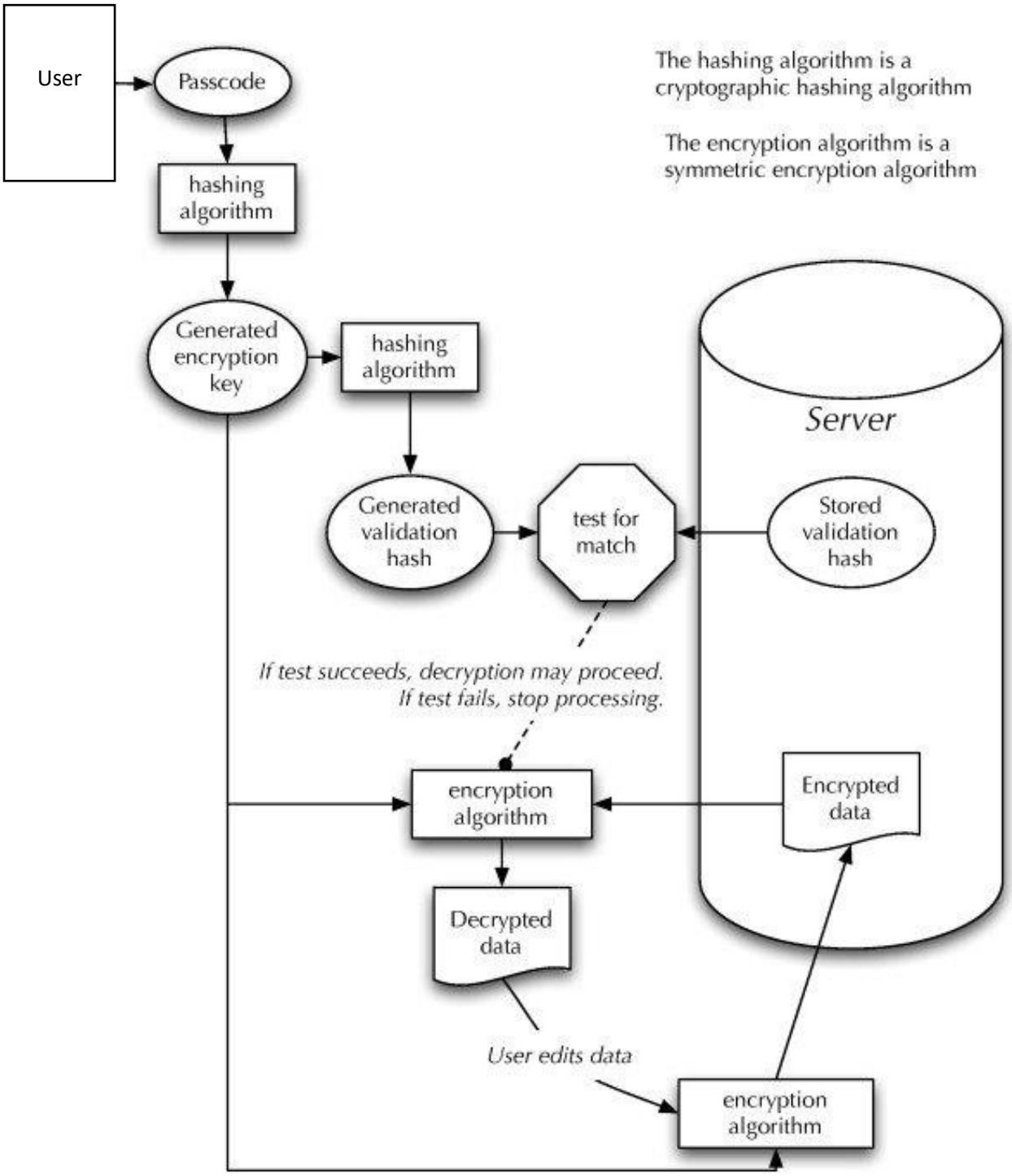
DFD (Data Flow Diagram)



Structure of Application



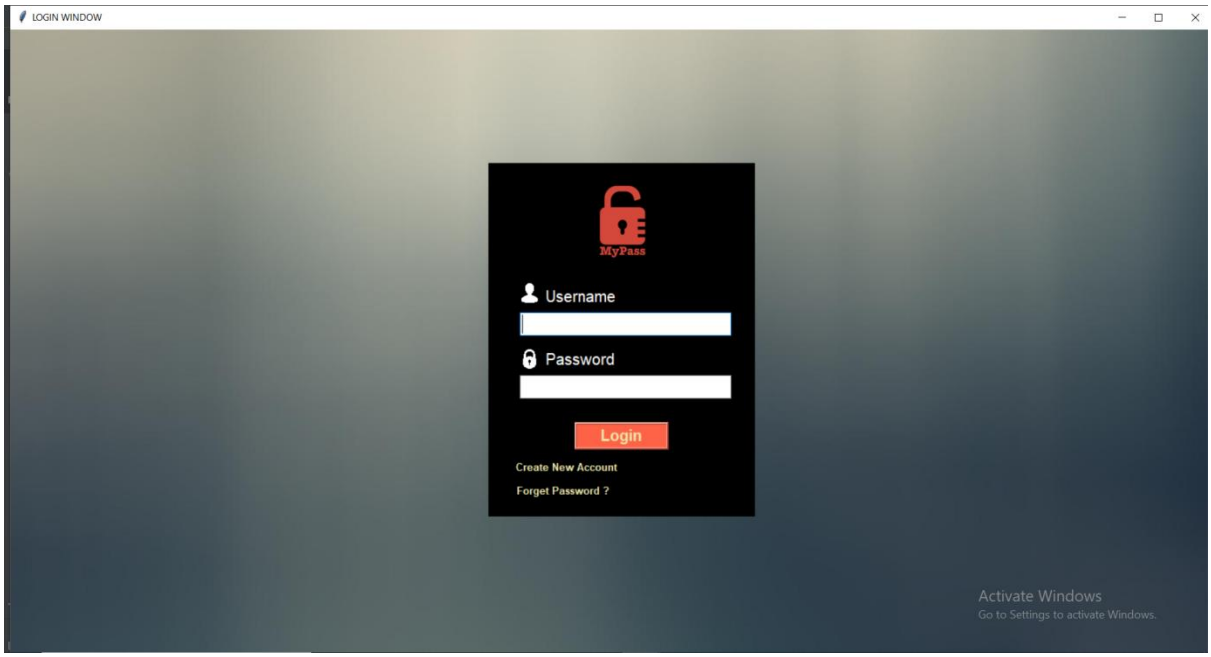
Entity Relationship Diagram



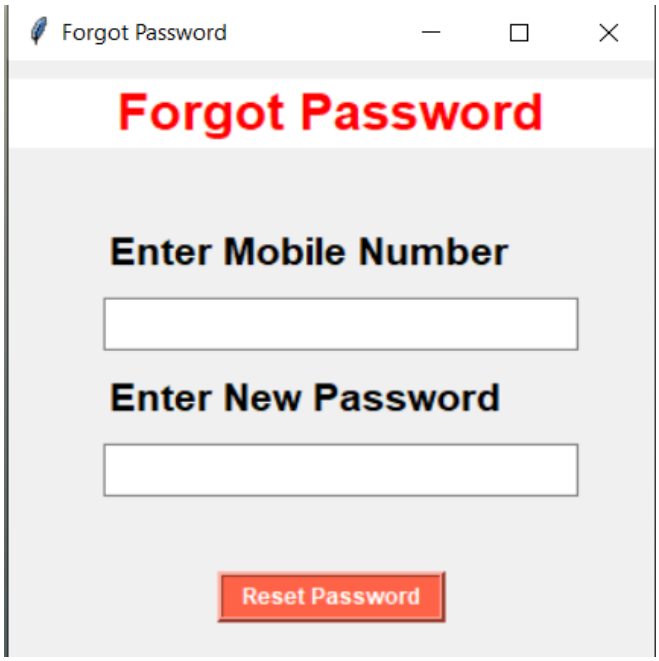
System Design

Form Design

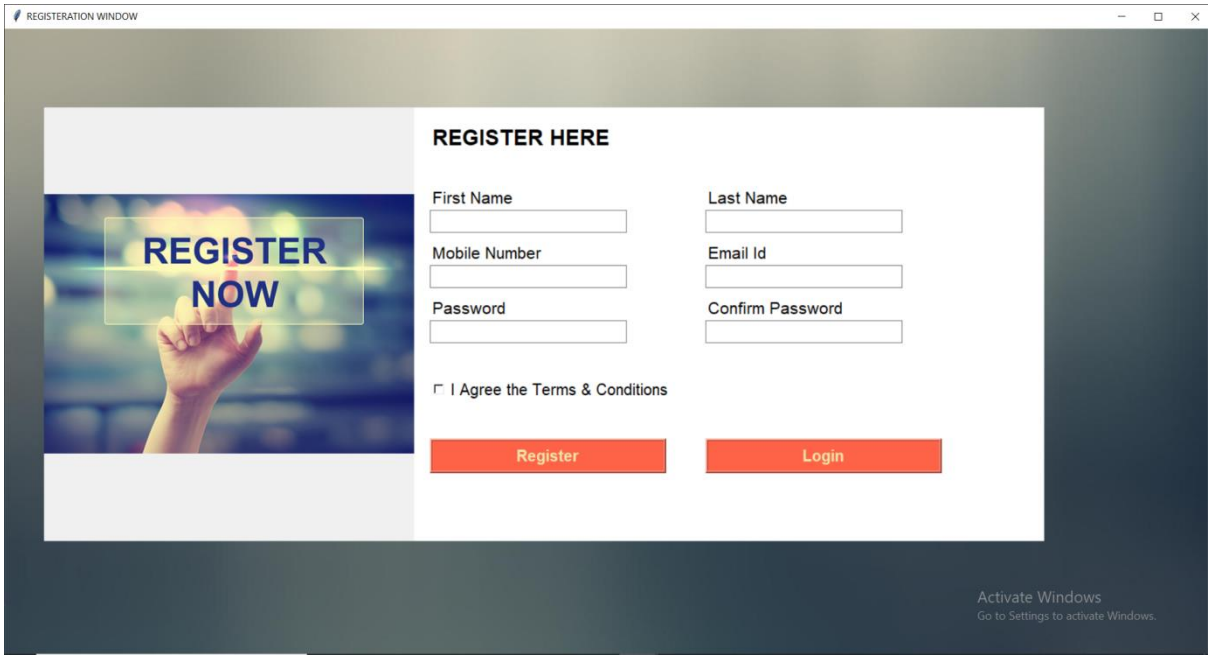
Login Page: This is the first screen of “ArchSafe” which directly takes the user to login page with an option to create a new account (Register) or if they have forgotten the password then to change it.



Forgot Password: This page checks mobile number of the user and changes their password accordingly if the number is correct and stored in database.



Registration Window: This page is where user gets the option to input their information which is to be stored in the database. This information comprises of their name, mobile number, email id, password, terms and conditions and an option to go back to login page.



Source Code

MAIN.PY

```
from tkinter import *
from PIL import Image,ImageTk
from tkinter import ttk
from tkinter import messagebox
import mysql.connector
import base64
import pyttsx3

from ciphers import RSAC, affineC, vernamOneTimePadC, caesarC, homophonicC, railfenceC, autokeyC,
atbashC, columnerC, \
   beaufortC, beaufortautokeyC, vignereC, vignereautokeyC, playfairC, morseC


class Login():
   def __init__(self,root):
      self.root=root
      self.root.title("LOGIN WINDOW")
      self.root.geometry("{0}x{1}+0+0".format(root.winfo_screenwidth(), root.winfo_screenheight()))
      self.bg=ImageTk.PhotoImage(file="bg.jpg")
      lab1_bg=Label(self.root,image=self.bg)
      lab1_bg.place(x=0,y=0,relwidth=1,relheight=1)

      frame=Frame(self.root,bg="black")
      frame.place(x=610,y=170,width=340,height=450)

      img1=Image.open("logo.png")
      img1=img1.resize((110,110),Image.ANTIALIAS)
      self.photoimage1=ImageTk.PhotoImage(img1)

      labimg1=Label(image=self.photoimage1,bg="black",borderwidth=0)
      labimg1.place(x=730,y=195,width=100,height=100)

      get_str=Label(frame,text="Login Page",font=("Arial",20,"bold"),fg="black",bg="black")
      get_str.place(x=95,y=100)

      #Label
      username=lb1=Label(frame,text="Username",font=("Arial",15),fg="white",bg="black")
      username.place(x=70,y=155)

      self.txtuser=ttk.Entry(frame,font=("Arial",15,"bold"))
      self.txtuser.place(x=40,y=190,width=270)

      password = lb1 = Label(frame, text="Password", font=("Arial", 15), fg="white", bg="black")
      password.place(x=70, y=235)

      self.txtpass = ttk.Entry(frame, font=("Arial", 15, "bold"))
      self.txtpass.place(x=40, y=270, width=270)
```

```
        # Icon Image
        img2 = Image.open("user.png")
        img2 = img2.resize((25, 25), Image.ANTIALIAS)
        self.photoimage2 = ImageTk.PhotoImage(img2)

        labimg1 = Label(image=self.photoimage2, bg="black", borderwidth=0)
        labimg1.place(x=650, y=323, width=25, height=25)

        img3 = Image.open("pass.png")
        img3 = img3.resize((25, 25), Image.ANTIALIAS)
        self.photoimage3 = ImageTk.PhotoImage(img3)

        labimg1 = Label(image=self.photoimage3, bg="black", borderwidth=0)
        labimg1.place(x=650, y=405, width=25, height=30)


        # Button
        loginbtn=Button(frame,text="Login",font=("Arial", 15,
"bold"),bd=3,relief=RIDGE,fg="#EEE8AA",bg="#FF6347",command=self.login)
        loginbtn.place(x=110,y=330,width=120,height=35)

        registerbtn = Button(frame, text="Create New Account",command=self.register_window,
font=("Arial", 10, "bold"), borderwidth=0, fg="#EEE8AA",bg="black")
        registerbtn.place(x=20, y=375, width=160)

        forgetbtn = Button(frame, text="Forget Password ?",command=self.forgot_pass, font=("Arial", 10,
"bold"), borderwidth=0, fg="#EEE8AA",bg="black")
        forgetbtn.place(x=15, y=405, width=160)

    def register_window(self):
        self.new_window=Toplevel(self.root)
        self.app=Register(self.new_window)


    def login(self):
        if self.txtuser.get()=="" or self.txtpass.get()=="":

            messagebox.showerror("Error","Username or Password cannot be empty.")
        elif self.txtuser.get()=="Kanha" and self.txtpass.get()=="Nachi":
            messagebox.showinfo("Success","Login Successful.")
        else:
            conn = mysql.connector.connect(host="localhost", user="root",
password="Testkanha123",database="archsafedb")
            my_cursor = conn.cursor()
            my_cursor.execute("select * from regtable where Email=%s and Password=%s",(
                self.txtuser.get(),
                self.txtpass.get()
            ))
            row=my_cursor.fetchone()
            if row==None:
                messagebox.showerror("Error","Invalid Username and Password")
            else:
                open_main=messagebox.askyesno("YesNo","Access only admin")
                if open_main>0:
                    wind1=Toplevel()
```

```
                app=project(wind1)

            else:
                if not open_main:
                    return

            conn.commit()
            conn.close()

    #Forgot Password
        def forgot_pass(self):
            if self.txtuser.get()=="":
                messagebox.showerror("Error","Please enter the email")
            else:
                conn = mysql.connector.connect(host="localhost", user="root", password="Testkanha123",
                                               database="archsafedb")
                my_cursor = conn.cursor()
                query=("select * from regtable where Email=%s")
                value=(self.txtuser.get(),)
                my_cursor.execute(query,value)
                row=my_cursor.fetchone()
                # print(row)

                if row==None:
                    messagebox.showerror("Error","Please Enter the valid Username")
                else:
                    conn.close()
                    self.root2=Toplevel()
                    self.root2.title("Forgot Password")
                    self.root2.geometry("340x450+610+170")

                    l=Label(self.root2,text="Forgot Password",font=("Arial", 20, "bold"),fg="red",bg="white")
                    l.place(x=0,y=10,relwidth=1)

                    new_mob=Label(self.root2,text="Enter Mobile Number",font=("Arial", 16, "bold"),fg="black")
                    new_mob.place(x=50,y=90)

                    self.new_mob_entry = ttk.Entry(self.root2, font=("Arial", 15))
                    self.new_mob_entry.place(x=50, y=130, width=250)

                    new_pass = Label(self.root2, text="Enter New Password", font=("Arial", 16, "bold"),
    fg="black")
                    new_pass.place(x=50, y=170)

                    self.new_pass_entry = ttk.Entry(self.root2, font=("Arial", 15))
                    self.new_pass_entry.place(x=50, y=210, width=250)


                    forgotbtn = Button(self.root2, text="Reset Password", font=("Arial", 9, "bold"), bd=3,
    relief=RIDGE, fg="white",
                                       bg="#FF6347",command=self.reset_pass)
                    forgotbtn.place(x=110, y=280, width=120)

        def reset_pass(self):
            if self.new_mob_entry.get()=="":
```

```
            messagebox.showerror("Error","Enter Registered Mobile Number")
        elif self.new_pass_entry.get()=="":
            messagebox.showerror("Error", "Enter New Password")

        else:
            conn = mysql.connector.connect(host="localhost", user="root", password="Testkanha123",
                                database="archsafedb")
            my_cursor = conn.cursor()
            query=("select * from regtable where Email=%s and Password=%s")
            value=(self.txtuser.get(),self.txtpass.get())
            my_cursor.execute(query,value)
            row=my_cursor.fetchone()
            if row!=None:
                messagebox.showerror("Error","Enter valid Information")
            else:
                query=("update regtable set Password=%s where Email=%s ")
                value=(self.new_pass_entry.get(),self.txtuser.get())
                my_cursor.execute(query,value)

                conn.commit()
                conn.close()
                messagebox.showinfo("Info","Your Password has been Reset,Please Login new Password")


class Register():
    def __init__(self,root):
        self.root = root
        self.root.title("REGISTERATION WINDOW")
        self.root.geometry("{0}x{1}+0+0".format(root.winfo_screenwidth(), root.winfo_screenheight()))

        # Variable
        self.var_fname=StringVar()
        self.var_lname = StringVar()
        self.var_contact = StringVar()
        self.var_email = StringVar()
        self.var_pass = StringVar()
        self.var_cpass = StringVar()
        self.bg = ImageTk.PhotoImage(file="bg.jpg")
        bg_lab1 = Label(self.root, image=self.bg)
        bg_lab1.place(x=0, y=0, relwidth=1, relheight=1)

        self.bg1 = ImageTk.PhotoImage(file="register.jpg")
        left_lab1 = Label(self.root, image=self.bg1)
        left_lab1.place(x=50, y=100,width=470,height=550)

        frame = Frame(self.root, bg="white")
        frame.place(x=520, y=100, width=800, height=550)

        # Label
        reg_lbl = Label(frame,text="REGISTER HERE",font=("Arial",20,"bold"),fg="black",bg="white")
        reg_lbl.place(x=20, y=20)

        # Row1
        # Firstname
        fname=Label(frame,text="First Name",font=("Arial",16),fg="black",bg="white")
```

```
        fname.place(x=20,y=100)

        self.fname_entry=ttk.Entry(frame,textvariable=self.var_fname,font=("Arial",15))
        self.fname_entry.place(x=20,y=130,width=250)

        # Lastname
        lname = Label(frame, text="Last Name", font=("Arial", 16), fg="black", bg="white")
        lname.place(x=370, y=100)

        self.lname_entry = ttk.Entry(frame,textvariable=self.var_lname, font=("Arial", 15))
        self.lname_entry.place(x=370, y=130, width=250)

        # Row2
        contact=Label(frame,text="Mobile Number",font=("Arial",16),fg="black",bg="white")
        contact.place(x=20,y=170)
        self.txt_contact = ttk.Entry(frame,textvariable=self.var_contact, font=("Arial", 15))
        self.txt_contact.place(x=20, y=200, width=250)

        email = Label(frame, text="Email Id", font=("Arial", 16), fg="black", bg="white")
        email.place(x=370, y=170)
        self.txt_email = ttk.Entry(frame,textvariable=self.var_email, font=("Arial", 15))
        self.txt_email.place(x=370, y=200, width=250)

        # Row3
        pwd = Label(frame, text="Password", font=("Arial", 16), fg="black", bg="white")
        pwd.place(x=20, y=240)
        self.txt_pwd = ttk.Entry(frame,textvariable=self.var_pass, font=("Arial", 15))
        self.txt_pwd.place(x=20, y=270, width=250)

        cpwd = Label(frame, text="Confirm Password", font=("Arial", 16), fg="black", bg="white")
        cpwd.place(x=370, y=240)
        self.txt_cpwd = ttk.Entry(frame,textvariable=self.var_cpass, font=("Arial", 15))
        self.txt_cpwd.place(x=370, y=270, width=250)

        #button
        self.var_check=IntVar()
        self.checkbtn=Checkbutton(frame,variable=self.var_check,text="I Agree the Terms &
Conditions",font=("Arial", 15), fg="black", bg="white",onvalue=1,offvalue=0)
        self.checkbtn.place(x=20,y=340)

        regbtn = Button(frame, text="Register", font=("Arial", 15, "bold"), bd=3, relief=RIDGE,
fg="#EEE8AA",bg="#FF6347",command=self.register_data)
        regbtn.place(x=20,y=420,width=300)

        loginbtn = Button(frame, text="Login", font=("Arial", 15, "bold"), bd=3, relief=RIDGE,
fg="#EEE8AA",bg="#FF6347")
        loginbtn.place(x=370, y=420, width=300)

    def register_data(self):
        if self.var_fname.get()=="" or self.var_lname.get()=="" or self.var_email.get()=="" or
self.var_contact.get()=="":
            messagebox.showerror("Error","All fields are required")

        elif self.var_pass.get() != self.var_cpass.get():
            messagebox.showerror("Error","Password and Confirm Password must be same.")
```

```
        elif self.var_check.get()==0:
            messagebox.showerror("Error", "Please agree our Terms & Conditions")
        else:

conn=mysql.connector.connect(host="localhost",user="root",password="Testkanha123",database="archsaf
edb")
            my_cursor=conn.cursor()
            query=("select * from regtable where email=%s")
            value=(self.var_email.get(),)
            my_cursor.execute(query,value)
            row=my_cursor.fetchone()
            if row!=None:
                messagebox.showerror("Error","User already exist , Please try another email")
            else:
                my_cursor.execute("insert into regtable values(%s,%s,%s,%s,%s)",(
                    self.var_fname.get(),
                    self.var_lname.get(),
                    self.var_contact.get(),
                    self.var_email.get(),
                    self.var_pass.get()
                ))
            conn.commit()
            conn.close()
            messagebox.showinfo("Success","Register Successfully")

def project(root):
    root = Tk()
    root.configure(background="gray")
    root.title("cryptociphers")
    title_frame = Frame(root, width=180, height=180, relief=SUNKEN, borderwidth=10)
    title_frame.pack()


    # img=Image.open('C:/Users/Kolluri Midhun/OneDrive/Documents/bennett.png')
    # img=ImageTk.PhotoImage(img)
    # icon=Button(title_frame,image=img)
    # icon.pack()


##############################################################################################
##########################################

    def cipher():
        root.destroy()

        window = Tk()

        window.configure(background='gray')
        # window.geometry('1000x600+100+50')
        window.title("Cryptographic ciphers")

##############################################################################################
##############################################
        # frame code
```

```
    left_frame = Frame(window, width=200, height=600, relief=SUNKEN)
    left_frame.pack(side=LEFT)
    right_frame = Frame(window, width=200, height=600, relief=SUNKEN)
    right_frame.pack(side=RIGHT)

    main_frame = Frame(window, width=800, height=100, relief=SUNKEN, borderwidth=10)
    main_frame.pack()

    main = Frame(window, width=800, height=400, relief=SUNKEN, bg='black')
    main.pack()

    time_frame = Frame(window, width=500, height=30, relief=SUNKEN, background='gray')
    time_frame.pack(side=BOTTOM)


##########################################################################################
########################################

    # img=Image.open('C:/Users/Kolluri Midhun/OneDrive/Pictures/title_image.png')
    # img=ImageTk.PhotoImage(img)
    # icon=Button(main_frame,image=img)
    # icon.pack()

    def remove():
        for widget in main.winfo_children():
            widget.destroy()

    def window_show():
        remove()

    def lab():

        text_label = Label(main, text="Enter text: ", font=('fixedsys', 16, "bold"), bg="tomato2",
fg="white")
        text_label.grid(row=0, column=0, padx=20, pady=20)

        scroll_text = ttk.Scrollbar(main, orient=VERTICAL)
        text_box = Text(main, height=8, width=40, pady=10, yscrollcommand=scroll_text.set, bg='white')
        text_box.grid(row=1, column=0, pady=1, padx=1)
        scroll_text.config(command=text_box.yview)
        scroll_text.grid(row=1, column=1, sticky='NS')

        key_label = Label(main, text="Enter key: ", font=('fixedsys', 14), pady=15,
                          bg='black', fg="cyan")
        key_label.grid(row=2, column=0)

        scroll_text2 = ttk.Scrollbar(main, orient=VERTICAL)
        new_text = Text(main, height=8, width=40, pady=10, yscrollcommand=scroll_text2.set,
bg='white')
        new_text.grid(row=1, column=2, columnspan=2, padx=(10, 0))
        scroll_text2.config(command=new_text.yview)
        scroll_text2.grid(row=1, column=4, sticky='NS')
        return text_box, new_text
```

```
##########################################################################################
##########################################
    def caesar_cipher():  # 1st cipher

        remove()

        # label.config(text = "Caesar Cipher")
        # label.grid(row = 0, column = 0)

        list_key = ttk.Combobox(main)
        list_key['values'] = (
        1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26)
        list_key.current(0)
        list_key.grid(row=5, column=0)

        text_box, new_text = lab()
        sample = "Sample text input:\nAny character\n\nSample Key input:\nnumerical value 1-
26\nDefault: 1"
        new_text.insert(1.0, sample)
        label = Label(main, text="caesar_cipher")
        label.grid(row=0, column=1)

        def encrypt():
            new_text.delete('1.0', END)
            txt = text_box.get("1.0", END)
            key = int(list_key.get())
            enc_text = caesarC.encryption(txt, key)
            new_text.insert(1.0, enc_text)

        enc = Button(main, text="Encrypt", bd=10, width=10, command=encrypt, bg='tomato2', fg='white')
        enc.grid(row=0, column=2, padx=20, pady=30)

        def decrypt():
            new_text.delete('1.0', END)
            txt = text_box.get("1.0", END)
            key = int(list_key.get())
            dec_text = caesarC.decryption(txt, key)
            new_text.insert(1.0, dec_text)

        dec = Button(main, text="Decrypt", bd=10, width=10, command=decrypt, bg='tomato2', fg='white')
        dec.grid(row=0, column=3, padx=10, pady=10)


##########################################################################################
########################################
    def homophonic_cipher():
        remove()

        key_table = {'A': 'D9', 'B': 'X', 'C': 'S', 'D': 'F', 'E': 'Z721', 'F': 'E', 'G': 'H', 'H': 'C', 'I': 'V3',
                     'J': 'I', 'K': 'T', 'L': 'P', 'M': 'G', 'N': 'A5', 'O': 'Q0', 'P': 'L', 'Q': 'K', 'R': 'J',
                     'S': 'R4', 'T': '6U',
                     'U': 'O', 'V': 'W', 'W': 'M', 'X': 'Y', 'Y': 'B', 'Z': 'N', 'a': 'd(', 'b': 'x', 'c': 's',
                     'd': 'f', 'e': 'z&@!',
                     'f': 'e', 'g': 'h', 'h': 'c', 'i': 'v#', 'j': 'i', 'k': 't', 'l': 'p', 'm': 'g', 'n': 'a%',
```

```
                'o': 'q)', 'p': 'l',
                'q': 'k', 'r': 'j', 's': 'r$', 't': '^u', 'u': 'o', 'v': 'w', 'w': 'm', 'x': 'y', 'y': 'b',
                'z': 'n'}

        text_label = Label(main, text="Enter text: ", font=('fixedsys', 12, 'bold'), bg='black', fg="cyan")
        text_label.grid(row=0, column=0, padx=20, pady=20)

        scroll_text = ttk.Scrollbar(main, orient=VERTICAL)
        label = Label(main, text="homophonic_cipher")
        label.grid(row=0, column=1)
        text_box = Text(main, height=20, width=40, pady=10, yscrollcommand=scroll_text.set,
bg='plum1')
        text_box.grid(row=1, column=0, pady=1, padx=1)
        scroll_text.config(command=text_box.yview)
        scroll_text.grid(row=1, column=1, sticky='NS')

        scroll_text2 = ttk.Scrollbar(main, orient=VERTICAL)
        new_text = Text(main, height=20, width=40, pady=10, yscrollcommand=scroll_text2.set,
bg='plum1')
        new_text.grid(row=1, column=2, columnspan=2, padx=(10, 0))
        sample = "Sample text input:\nAny character\n"
        new_text.insert(1.0, sample)

        scroll_text2.config(command=new_text.yview)
        scroll_text2.grid(row=1, column=4, sticky='NS')

        def encrypt():
            new_text.delete('1.0', END)
            txt = text_box.get("1.0", END)
            enc_text = homophonicC.encryption(txt)
            new_text.insert(1.0, enc_text)

        enc = Button(main, text="Encrypt", bd=10, width=10, command=encrypt,
                     bg='tomato2', fg='white')
        enc.grid(row=0, column=2, padx=20, pady=20)

        def decrypt():
            new_text.delete('1.0', END)
            txt = text_box.get("1.0", END)
            dec_text = homophonicC.decryption(txt)
            new_text.insert(1.0, dec_text)

        dec = Button(main, text="Decrypt", bd=10, width=10, command=decrypt,
                     bg='tomato2', fg='white')
        dec.grid(row=0, column=3, padx=10, pady=10)


##########################################################################################
########################################
    def vignere_cipher():
        remove()

        key_text = Entry(main, width=40)
        key_text.grid(row=3, column=0, padx=10, pady=10)
```

```
        text_box, new_text = lab()
        sample = "Sample text input:\nAny character\n\nSample Key input:\nAlphabet/Word\nDefault
Key: a"
        new_text.insert(1.0, sample)
        label = Label(main, text="vignere_cipher")
        label.grid(row=0, column=1)

        def encrypt():
            new_text.delete('1.0', END)
            txt = text_box.get("1.0", END)
            key = key_text.get()
            enc_text = vignereC.encrypt(txt, key)
            new_text.insert(1.0, enc_text)

        enc = Button(main, text="Encrypt", bd=10, width=10, command=encrypt,
                     bg='tomato2', fg='white')
        enc.grid(row=0, column=2, padx=20, pady=30)

        def decrypt():
            new_text.delete('1.0', END)
            txt = text_box.get("1.0", END)
            key = key_text.get()
            dec_text = vignereC.decrypt(txt, key)
            new_text.insert(1.0, dec_text)

        dec = Button(main, text="Decrypt", bd=10, width=10, command=decrypt,
                     bg='tomato2', fg='white')
        dec.grid(row=0, column=3, padx=10, pady=10)


##########################################################################################
########################################
    def autokey_cipher():
        remove()

        key_text = Entry(main, width=40)
        key_text.grid(row=3, column=0, padx=10, pady=10)

        text_box, new_text = lab()
        sample = "Sample text input:\nAny character\n\nSample Key input:\nAlphabet/Word\nDefault: A"
        new_text.insert(1.0, sample)
        label = Label(main, text="autokey_cipher")
        label.grid(row=0, column=1)

        def encrypt():
            new_text.delete('1.0', END)
            txt = text_box.get("1.0", END)
            key = key_text.get()
            enc_text = autokeyC.encrypt(txt, key)
            new_text.insert(1.0, enc_text)

        enc = Button(main, text="Encrypt", bd=10, width=10, command=encrypt,
                     bg='tomato2', fg='white')
        enc.grid(row=0, column=2, padx=20, pady=30)
```

```
        def decrypt():
            new_text.delete('1.0', END)
            txt = text_box.get("1.0", END)
            key = key_text.get()
            dec_text = autokeyC.decrypt(txt, key)
            new_text.insert(1.0, dec_text)

        dec = Button(main, text="Decrypt", bd=10, width=10, command=decrypt,
                     bg='yellow', fg='black')
        dec.grid(row=0, column=3, padx=10, pady=10)


##########################################################################################
########################################

    def railfence_cipher():
        remove()

        list_key = ttk.Combobox(main)
        list_key['values'] = (2, 3, 4, 5, 6, 7, 8)
        list_key.current(0)
        list_key.grid(row=5, column=0)

        text_box, new_text = lab()
        sample = "Sample text input:\nAny character\n\nSample Key input:\nPositive integer < length of
input\nDefault: 2"
        new_text.insert(1.0, sample)
        label = Label(main, text="railfence_cipher")
        label.grid(row=0, column=1)

        def encrypt():
            new_text.delete('1.0', END)
            string = (text_box.get("1.0", END)).strip()
            key = int(list_key.get())
            enc_text = railfenceC.encrypt(string, key)
            new_text.insert(1.0, enc_text)

        enc = Button(main, text="Encrypt", bd=10, width=10, command=encrypt,
                     bg='tomato2', fg='white')
        enc.grid(row=0, column=2, padx=20, pady=30)

        def decrypt():
            new_text.delete('1.0', END)
            string = (text_box.get("1.0", END)).strip()
            key = int(list_key.get())
            dec_text = railfenceC.decrypt(string, key)
            new_text.insert(1.0, dec_text)

        dec = Button(main, text="Decrypt", bd=10, width=10, command=decrypt,
                     bg='tomato2', fg='white')
        dec.grid(row=0, column=3, padx=10, pady=10)


##########################################################################################
########################################
```

```
    def playfair_cipher():
        remove()

        key_text = Entry(main, width=40)
        key_text.grid(row=3, column=0, padx=10, pady=10)

        text_box, new_text = lab()
        sample = "Sample text input:\nAny character\n\nSample Key input:\nAlphabet/Word\nDefault:
ABCD\n(Note:Case of key & text should be same)"
        new_text.insert(1.0, sample)
        label = Label(main, text="playfair_cipher")
        label.grid(row=0, column=1)

        def encrypt():
            new_text.delete('1.0', END)
            txt = text_box.get("1.0", END)
            key = key_text.get()
            enc_text = playfairC.encrypt(txt, key)
            new_text.insert(1.0, enc_text)

        enc = Button(main, text="Encrypt", bd=10, width=10, command=encrypt,
                     bg='tomato2', fg='white')
        enc.grid(row=0, column=2, padx=20, pady=30)

        def decrypt():
            new_text.delete('1.0', END)
            txt = text_box.get("1.0", END)
            key = key_text.get()
            dec_text = playfairC.decrypt(txt, key)
            new_text.insert(1.0, dec_text)

        dec = Button(main, text="Decrypt", bd=10, width=10, command=decrypt,
                     bg='tomato2', fg='white')
        dec.grid(row=0, column=3, padx=10, pady=10)


##########################################################################################
########################################

    def atbash_cipher():
        remove()

        text_label = Label(main, text="Enter text: ", font=('fixedsys', 12, 'bold'), bg='black', fg="cyan")
        text_label.grid(row=0, column=0, padx=20, pady=20)

        scroll_text = ttk.Scrollbar(main, orient=VERTICAL)
        label = Label(main, text="atbash_cipher")
        label.grid(row=0, column=1)
        text_box = Text(main, height=20, width=40, pady=10, yscrollcommand=scroll_text.set,
bg='plum1')
        text_box.grid(row=1, column=0, pady=1, padx=1)
        scroll_text.config(command=text_box.yview)
        scroll_text.grid(row=1, column=1, sticky='NS')
```

```
        scroll_text2 = ttk.Scrollbar(main, orient=VERTICAL)
        new_text = Text(main, height=20, width=40, pady=10, yscrollcommand=scroll_text2.set,
bg='plum1')
        new_text.grid(row=1, column=2, columnspan=2, padx=(10, 0))
        scroll_text2.config(command=new_text.yview)
        scroll_text2.grid(row=1, column=4, sticky='NS')
        sample = "Sample text input:\nAny character\n"
        new_text.insert(1.0, sample)

        def convert():
            new_text.delete('1.0', END)
            string = text_box.get("1.0", END)
            converted_text = atbashC.encrypt(string)
            new_text.insert(1.0, converted_text)

        enc = Button(main, text="Encrypt", bd=10, width=10, command=convert,
                     bg='tomato2', fg='white')
        enc.grid(row=0, column=2, padx=20, pady=10)

        dec = Button(main, text="Decrypt", bd=10, width=10, command=convert,
                     bg='tomato2', fg='white')
        dec.grid(row=0, column=3, padx=10, pady=20)


##########################################################################################
############################

    def vignere_autokey_cipher():
        remove()

        key_text = Entry(main, width=40)
        key_text.grid(row=3, column=0, padx=10, pady=10)

        text_box, new_text = lab()
        sample = "Sample text input:\nAny character\n\nSample Key input:\nAlphabet/Word\nDefault
Key: a"
        new_text.insert(1.0, sample)
        label = Label(main, text="vignere_autokey_cipher")
        label.grid(row=0, column=1)

        def encrypt():
            new_text.delete('1.0', END)
            txt = text_box.get("1.0", END)
            key = key_text.get()
            enc_text = vignereautokeyC.encrypt(txt, key)
            new_text.insert(1.0, enc_text)

        enc = Button(main, text="Encrypt", bd=10, width=10, command=encrypt,
                     bg='tomato2', fg='white')
        enc.grid(row=0, column=2, padx=20, pady=30)

        def decrypt():
            new_text.delete('1.0', END)
            txt = text_box.get("1.0", END)
            key = key_text.get()
```

```
            dec_text = vignereautokeyC.decrypt(txt, key)
            new_text.insert(1.0, dec_text)

        dec = Button(main, text="Decrypt", bd=10, width=10, command=decrypt,
                     bg='tomato2', fg='white')
        dec.grid(row=0, column=3, padx=10, pady=10)


##########################################################################################
########################################
    def beaufort_cipher():
        remove()

        key_text = Entry(main, width=40)
        key_text.grid(row=3, column=0, padx=10, pady=10)

        text_box, new_text = lab()
        sample = "Sample text input:\nAny character\n\nSample Key input:\nAlphabet/Word\nDefault
Key: abc"
        new_text.insert(1.0, sample)
        label = Label(main, text="beaufort_cipher")
        label.grid(row=0, column=1)

        def convert():
            spl = ['!', '@', '#', '$', '%', '^', '&', '*', '(', ')', '<', '~', ':', ';', '<', '>', '?', '/']
            new_text.delete('1.0', END)
            txt = text_box.get("1.0", END)
            key = key_text.get()
            converted_text = beaufortC.convert(txt, key)
            new_text.insert(1.0, converted_text)

        enc = Button(main, text="Encrypt", bd=10, width=10, command=convert,
                     bg='tomato2', fg='white')
        enc.grid(row=0, column=2, padx=20, pady=10)

        dec = Button(main, text="Decrypt", bd=10, width=10, command=convert,
                     bg='tomato2', fg='white')
        dec.grid(row=0, column=3, padx=10, pady=30)


##########################################################################################
########################################
    def beaufort_autokey_cipher():
        remove()

        key_text = Entry(main, width=40)
        key_text.grid(row=3, column=0, padx=10, pady=10)

        text_box, new_text = lab()
        sample = "Sample text input:\nAny character\n\nSample Key input:\nAlphabet/Word\nDefault
Key: abc"
        new_text.insert(1.0, sample)
        label = Label(main, text="beaufort_autokey_cipher")
        label.grid(row=0, column=1)
```

```
        def encrypt():
            new_text.delete('1.0', END)
            txt = text_box.get("1.0", END)
            key = key_text.get()
            converted_text = beaufortautokeyC.encrypt(txt, key)
            new_text.insert(1.0, converted_text)

        def decrypt():
            new_text.delete('1.0', END)
            txt = text_box.get("1.0", END)
            key = key_text.get()
            converted_text = beaufortautokeyC.decrypt(txt, key)
            new_text.insert(1.0, converted_text)

        enc = Button(main, text="Encrypt", bd=10, width=10, command=encrypt,
                     bg='tomato2', fg='white')
        enc.grid(row=0, column=2, padx=20, pady=30)

        dec = Button(main, text="Decrypt", bd=10, width=10, command=decrypt,
                     bg='tomato2', fg='white')
        dec.grid(row=0, column=3, padx=10, pady=10)


########################################################################################################
#####################################
    def columnar_trans_cipher():
        remove()

        key_text = Entry(main, width=40)
        key_text.grid(row=3, column=0, padx=10, pady=10)

        text_box, new_text = lab()
        sample = "Sample text input:\nAny character\n\nSample Key input:\nAlphabet/Word\nDefault
Key: key"
        new_text.insert(1.0, sample)
        label = Label(main, text="columnar_trans_cipher")
        label.grid(row=0, column=1)

        def encrypt():
            new_text.delete('1.0', END)
            txt = text_box.get("1.0", END)
            key = key_text.get().lower()
            enc_text = columnerC.encrypt(txt, key)
            new_text.insert(1.0, enc_text)

        enc = Button(main, text="Encrypt", bd=10, width=10, command=encrypt,
                     bg='tomato2', fg='white')
        enc.grid(row=0, column=2, padx=20, pady=30)

        def decrypt():
            new_text.delete('1.0', END)
            txt = text_box.get("1.0", END)
            key = key_text.get().lower()
            dec_text = columnerC.decrypt(txt, key)
            new_text.insert(1.0, dec_text)
```

```
        dec = Button(main, text="Decrypt", bd=10, width=10, command=decrypt,
                     bg='tomato2', fg='white')
        dec.grid(row=0, column=3, padx=10, pady=10)


##########################################################################################
########################################
    def morse_code():

        remove()

        text_label = Label(main, text="Enter text: ", font=('fixedsys', 12, 'bold'), bg='black', fg="cyan")
        text_label.grid(row=0, column=0, padx=20, pady=20)

        scroll_text = ttk.Scrollbar(main, orient=VERTICAL)
        label = Label(main, text="morse_code_cipher")
        label.grid(row=0, column=1)
        text_box = Text(main, height=20, width=40, pady=10, yscrollcommand=scroll_text.set,
bg='plum1')
        text_box.grid(row=1, column=0, pady=1, padx=1)
        scroll_text.config(command=text_box.yview)
        scroll_text.grid(row=1, column=1, sticky='NS')

        scroll_text2 = ttk.Scrollbar(main, orient=VERTICAL)
        new_text = Text(main, height=20, width=40, pady=10, yscrollcommand=scroll_text2.set,
bg='plum1')
        new_text.grid(row=1, column=2, columnspan=2, padx=(10, 0))
        scroll_text2.config(command=new_text.yview)
        scroll_text2.grid(row=1, column=4, sticky='NS')
        sample = "Sample text input:\nCapital Letter/Numbers/Punctuations\n\nFor decryption: Add spaces
between each character"
        new_text.insert(1.0, sample)

        Morse_dict = {
           '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': '--..',
           '1': '.----', '2': '..---', '3': '...--', '4': '....-', '5': '.....', '6': '-....',
           '7': '--...', '8': '---..', '9': '----.', '0': '-----', ',': '--..--', '.': '.-.-.-',
           '?': '..--..', '/': '-..-.', '-': '-....-', '(': '-.--.', ')': '-.--.-', ' ': '`'}

        Reverse_morse = {val: key for key, val in Morse_dict.items()}

        def encrypt():
           new_text.delete('1.0', END)
           txt = text_box.get("1.0", END)
           enc_text = morseC.encrypt(txt)
           new_text.insert(1.0, enc_text)

        enc = Button(main, text="Encrypt", bd=10, width=10, command=encrypt,
                     bg='tomato2', fg='white')
        enc.grid(row=0, column=2, padx=20, pady=20)
```

```
        def decrypt():
            new_text.delete('1.0', END)
            txt = text_box.get('1.0', END)
            dec_text = morseC.decrypt(txt)
            new_text.insert(1.0, dec_text)

        dec = Button(main, text='Decrypt', bd=10, width=10, command=decrypt,
                     bg='tomato2', fg='white')
        dec.grid(row=0, column=3, padx=10, pady=10)


##########################################################################################
########################################
    def onetimepad():
        remove()

        key_text = Text(main, height=1, width=10, pady=5, bg='white')
        key_text.grid(row=3, column=0, pady=1, padx=1)

        text_box, new_text = lab()
        sample = "Sample text input:\nAny character\n\nInput Type\n for alphabets 1\n for binary 2 \n for
numbers 3"
        new_text.insert(1.0, sample)
        text_label = Label(main, text="Input Type: ", font=('fixedsys', 12, 'bold'), bg='black', fg="cyan")
        text_label.grid(row=4, column=0, pady=5)
        box = Text(main, height=1, width=10, pady=5, bg='white')
        box.grid(row=5, column=0, pady=1, padx=1)
        label = Label(main, text="onetimepad_cipher")
        label.grid(row=0, column=1)

        def encrypt():

            new_text.delete('1.0', END)
            plainText = text_box.get("1.0", END)
            key = key_text.get('1.0', END)

            input_type = box.get(1.0, END)

            def encryption(plainText, key, input_type):

                method = [26, 2, 10]
                subract = [97, 48, 48]

                input_type = int(input_type)
                plainText, key = plainText.rstrip().lower().split(), key.rstrip().lower().split()
                cipherText = []

                for i in range(len(plainText)):
                    partial_cipherText = ""
                    for j in range(len(plainText[i])):
                        c = (ord(plainText[i][j]) + ord(key[i][j]) - 2 * subract[input_type - 1]) % method[
                            input_type - 1]
                        partial_cipherText += chr(c + subract[input_type - 1])
                    cipherText.append(partial_cipherText)
```

```
            cipherText = " ".join(cipherText)

            enc_text = cipherText
            new_text.insert('1.0', enc_text)

        encryption(plainText, key, input_type)

    enc = Button(main, text="Encrypt", bd=10, width=10, command=encrypt, bg='tomato2', fg='white')
    enc.grid(row=0, column=2, padx=20, pady=30)

    def decrypt():

        new_text.delete('1.0', END)
        plainText = text_box.get("1.0", END)
        key = key_text.get('1.0', END)

        input_type = box.get(1.0, END)

        def decryption(cipherText, key, input_type):
            method = [26, 2, 10]
            subract = [97, 48, 48]
            input_type = int(input_type)

            cipherText, key = cipherText.lower().split(), key.lower().split()
            plainText = []

            for i in range(len(cipherText)):
                partial_plainText = ""
                for j in range(len(cipherText[i])):
                    p = (ord(cipherText[i][j]) - ord(key[i][j])) % method[input_type - 1]
                    partial_plainText += chr(p + subract[input_type - 1])
                plainText.append(partial_plainText)

            plainText = " ".join(plainText)

            enc_text = plainText
            new_text.insert('1.0', enc_text)

        decryption(plainText, key, input_type)

    dec = Button(main, text="Decrypt", bd=10, width=10, command=decrypt, bg='tomato2', fg='white')
    dec.grid(row=0, column=3, padx=10, pady=10)


##########################################################################################
#############################################

    def affine_cipher():

        remove()

        list_key = ttk.Combobox(main)
        list_key['values'] = (1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25)
        list_key.current(0)
        list_key.grid(row=5, column=0)
```

```
        list_key1 = ttk.Combobox(main)
        list_key1['values'] = (1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 15, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26)
        list_key1.current(0)
        list_key1.grid(row=6, column=0)

        text_box, new_text = lab()
        sample = "Sample text input:\nAny character\n\nSample Key input:\nnumerical value 1-
26\nDefault: 1"
        new_text.insert(1.0, sample)
        label = Label(main, text="Affine_cipher")
        label.grid(row=0, column=1)

        def encrypt():
            new_text.delete('1.0', END)
            txt = text_box.get("1.0", END)
            key = int(list_key.get())
            key1 = int(list_key1.get())
            enc_text = affineC.encryption(txt, key, key1)
            new_text.insert('1.0', enc_text)

        enc = Button(main, text="Encrypt", bd=10, width=10, command=encrypt, bg='tomato2', fg='white')
        enc.grid(row=0, column=2, padx=20, pady=30)

        def decrypt():
            new_text.delete('1.0', END)
            txt = text_box.get("1.0", END)
            key = int(list_key.get())
            key1 = int(list_key1.get())
            dec_text = affineC.decryption(txt, key, key1)
            new_text.insert('1.0', dec_text)

        dec = Button(main, text="Decrypt", bd=10, width=10, command=decrypt, bg='tomato2', fg='white')
        dec.grid(row=0, column=3, padx=10, pady=10)


##########################################################################################
#####################################
    # buttons on window left lide
    home_button = Button(left_frame, padx=20, bd=10, text='Home', width=20, height=3,
command=window_show,
                         bg='magenta4', fg='white', activebackground='black', font=('arial', 12, 'bold'),
                         activeforeground='SeaGreen1')
    home_button.grid(row=0, column=0)

    caesar = Button(left_frame, padx=20, bd=10, text='Caesar Cipher', width=20, height=3,
command=caesar_cipher,
                    bg='white', fg='red', activebackground='black', font=('arial', 12, 'bold'),
                    activeforeground='SeaGreen1')
    caesar.grid(row=1, column=0)

    homophonic = Button(left_frame, padx=20, bd=10, text='Homophonic Cipher', width=20, height=3,
                        command=homophonic_cipher, bg='white', fg='red', activebackground='black',
                        font=('arial', 12, 'bold'), activeforeground='SeaGreen1')
    homophonic.grid(row=2, column=0)
```

```
    vignere = Button(left_frame, padx=20, bd=10, text='Vignere Cipher', width=20, height=3,
command=vignere_cipher,
                     bg='white', fg='red', activebackground='black', font=('arial', 12, 'bold'),
                     activeforeground='SeaGreen1')
    vignere.grid(row=3, column=0)

    autokey = Button(left_frame, padx=20, bd=10, text='Autokey Cipher', width=20, height=3,
command=autokey_cipher,
                     bg='white', fg='red', activebackground='black', font=('arial', 12, 'bold'),
                     activeforeground='SeaGreen1')
    autokey.grid(row=4, column=0)

    railfence = Button(left_frame, padx=20, bd=10, text='Railfence Cipher', width=20, height=3,
                       command=railfence_cipher, bg='white', fg='red', activebackground='black',
                       font=('arial', 12, 'bold'), activeforeground='SeaGreen1')
    railfence.grid(row=5, column=0)

    affine = Button(left_frame, padx=20, bd=10, text='Affine Cipher', width=20, height=3,
command=affine_cipher,
                    bg='white', fg='red', activebackground='black', font=('arial', 12, 'bold'),
                    activeforeground='SeaGreen1')
    affine.grid(row=7, column=0)

    onetimepad = Button(left_frame, padx=20, bd=10, text='One Time Pad', width=20, height=3,
command=onetimepad,
                        bg='white', fg='red', font=('arial', 12, 'bold'), activebackground='black',
                        activeforeground='SeaGreen1')
    onetimepad.grid(row=8, column=0)

    playfair = Button(right_frame, padx=20, bd=10, text='Playfair Cipher', width=20, height=3,
command=playfair_cipher,
                      bg='white', fg='red', activebackground='black', font=('arial', 12, 'bold'),
                      activeforeground='SeaGreen1')
    playfair.grid(row=6, column=0)

    atbash = Button(right_frame, padx=20, bd=10, text='Atbash Cipher', width=20, height=3,
command=atbash_cipher,
                    bg='white', fg='red', activebackground='black', font=('arial', 12, 'bold'),
                    activeforeground='SeaGreen1')
    atbash.grid(row=9, column=0)

    vignere_autokey = Button(right_frame, padx=20, bd=10, text='Vignere Autokey Cipher', width=20,
height=3,
                             command=vignere_autokey_cipher, bg='white', fg='red', font=('arial', 12, 'bold'),
                             activebackground='black', activeforeground='yellow')
    vignere_autokey.grid(row=10, column=0)

    beaufort = Button(right_frame, padx=20, bd=10, text='Beaufort Cipher', width=20, height=3,
command=beaufort_cipher,
                      bg='white', fg='red', activebackground='black', font=('arial', 12, 'bold'),
                      activeforeground='SeaGreen1')
    beaufort.grid(row=11, column=0)

    beaufort_autokey = Button(right_frame, padx=20, bd=10, text='Beaufort Autokey Cipher', width=20,
height=3,
```

```
                    command=beaufort_autokey_cipher, bg='white', fg='red', font=('arial', 12, 'bold'),
                    activebackground='black', activeforeground='SeaGreen1')
    beaufort_autokey.grid(row=12, column=0)

    columnar = Button(right_frame, padx=20, bd=10, text='Columnar Transposition Cipher', width=20,
height=3,
                  command=columnar_trans_cipher, bg='white', fg='red', font=('arial', 12, 'bold'),
                  activebackground='black', activeforeground='SeaGreen1')
    columnar.grid(row=13, column=0)

    morse = Button(right_frame, padx=20, bd=10, text='Morse Code', width=20, height=3,
command=morse_code, bg='white',
                fg='red', font=('arial', 12, 'bold'), activebackground='black', activeforeground='SeaGreen1')
    morse.grid(row=14, column=0)
    Exit = Button(right_frame, text="Exit", bd=10, width=10, command=window.destroy,
activebackground="black",
                bg='magenta4', fg="white", font=('arial', 12, 'bold'), activeforeground="green2")
    Exit.grid(row=15, column=0)

    window.mainloop()



##########################################################################################
#####################################
  def hackertools():
    root.destroy()
    window = Tk()
    window.title("hackertools")
    label = Label(window, text=" hackertools").pack()
    left_frame = Frame(window, width=200, height=600, relief=SUNKEN)
    left_frame.pack(side=LEFT)
    main = Frame(window, width=200, height=600, relief=SUNKEN)
    main.pack()

    def remove():
      for widget in main.winfo_children():
        widget.destroy()

    def window_show():
      remove()
      downLabelName = Label(main, text=downLabel, fg="cyan", bg='black', font=('fixedsys', 17),
anchor=CENTER,
                    wraplength=250)
      downLabelName.grid(row=1, column=0)

    def lab():
      text_label = Label(main, text="Enter text: ", font=('fixedsys', 16, "bold"), bg="tomato2",
fg="white")
      text_label.grid(row=0, column=0, padx=20, pady=20)

      scroll_text = ttk.Scrollbar(main, orient=VERTICAL)
      text_box = Text(main, height=13, width=40, pady=10, yscrollcommand=scroll_text.set, bg='white')
      text_box.grid(row=1, column=0, pady=1, padx=1)
      scroll_text.config(command=text_box.yview)
```

```
        scroll_text.grid(row=1, column=1, sticky='NS')

        key_label = Label(main, text="Enter key: ", font=('fixedsys', 14), pady=15,
                          bg='black', fg="cyan")
        key_label.grid(row=2, column=0)

        scroll_text2 = ttk.Scrollbar(main, orient=VERTICAL)
        new_text = Text(main, height=13, width=40, pady=10, yscrollcommand=scroll_text2.set,
bg='white')
        new_text.grid(row=1, column=2, columnspan=2, padx=(10, 0))
        scroll_text2.config(command=new_text.yview)
        scroll_text2.grid(row=1, column=4, sticky='NS')
        return text_box, new_text

##########################################################################################
#######################################

    def caesar_cipher():
        # 1st cipher

        remove()

        # label.config(text = "Caesar Cipher")
        # label.grid(row = 0, column = 0)

        list_key = ttk.Combobox(main)
        list_key['values'] = (
        1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26)
        list_key.current(0)
        list_key.grid(row=5, column=0)

        text_box, new_text = lab()
        sample = "Sample text input:\nAny character\n\nSample Key input:\nnumerical value 1-
26\nDefault: 1"
        new_text.insert(1.0, sample)

        def encrypt():
            new_text.delete('1.0', END)
            txt = text_box.get("1.0", END)
            key = int(list_key.get())
            enc_text = caesarC.hacking(txt)
            new_text.insert(1.0, enc_text)

        enc = Button(main, text="hacking", bd=10, width=10, command=encrypt, bg='tomato2', fg='white')
        enc.grid(row=0, column=2, padx=20, pady=30)


##########################################################################################
#######################################

    def RSA():
        remove()

        list_key = ttk.Combobox(main)
        list_key['values'] = (
```

```
        1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26)
        list_key.current(0)
        list_key.grid(row=5, column=0)

        list_key1 = ttk.Combobox(main)
        list_key1['values'] = (
        1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26)
        list_key1.current(0)
        list_key1.grid(row=6, column=0)

        text_box, new_text = lab()
        sample = "Sample text input:\nAny character\n\nSample public Key input:\nnumerical value 1-
26\nDefault: 1"
        new_text.insert(1.0, sample)

        def encrypt():
            new_text.delete('1.0', END)
            txt = text_box.get("1.0", END)
            key = int(list_key.get())
            key1 = int(list_key1.get())
            enc_text = RSAC.encrypt((key, key1), txt)
            new_text.insert(1.0, enc_text)

        enc = Button(main, text="Encrypt", bd=10, width=10, command=encrypt, bg='tomato2', fg='white')
        enc.grid(row=0, column=2, padx=20, pady=30)

        def decrypt():
            new_text.delete('1.0', END)
            txt = text_box.get("1.0", END)
            key = int(list_key.get())
            key1 = int(list_key1.get())
            dec_text = RSAC.decrypt((key, key1), txt)
            new_text.insert(1.0, dec_text)

        dec = Button(main, text="Decrypt", bd=10, width=10, command=decrypt, bg='tomato2', fg='white')
        dec.grid(row=0, column=3, padx=10, pady=10)

        def hacking():
            new_text.delete('1.0', END)
            txt = text_box.get("1.0", END)
            key = int(list_key.get())
            key1 = int(list_key1.get())
            dec_text = RSAC.hacking((key, key1), txt)
            new_text.insert(1.0, dec_text)

        dec = Button(main, text="Hacking", bd=10, width=10, command=hacking, bg='tomato2',
fg='white')
        dec.grid(row=0, column=4, padx=10, pady=10)

    caesar = Button(left_frame, padx=20, bd=10, text='Caesar Cipher', width=20, height=3,
command=caesar_cipher,
                    bg='white', fg='red', activebackground='black', font=('arial', 12, 'bold'),
                    activeforeground='SeaGreen1')
    caesar.grid(row=1, column=0)
```

```
        caesar = Button(left_frame, padx=20, bd=10, text='RSA', width=20, height=3, command=RSA,
bg='white', fg='red',
                        activebackground='black', font=('arial', 12, 'bold'), activeforeground='SeaGreen1')
        caesar.grid(row=2, column=0)
        window.mainloop()



#################################################################################################
##############################


    Welcome_label = Label(root, text="Welcome to Cryptography ciphers", padx=10, pady=30, bg="light
gray", fg="black",
                          font=("Helvetica", 20, "bold")).pack(pady=30)
    chiper_button = Button(root, text="Cryptocypher", padx=20, bd=10, width=20, height=3,
command=cipher, bg="lightgreen",
                           fg="darkblue", activebackground='black', activeforeground='SeaGreen1')
    chiper_button.pack(pady=30)
    hacker_button = Button(root, text="Hackertools", padx=20, bd=10, width=20, height=3,
command=hackertools,
                           bg="lightgreen", fg="darkblue", activebackground='black',
activeforeground='SeaGreen1')

    hacker_button.pack()

    root.mainloop()




def main():
    win=Tk()
    app=Login(win)
    win.mainloop()




if __name__ == '__main__':
    main()
```

AffineC.py

```
def encryption(plainText, a, b):
    plainText = plainText.rstrip()

    cipherText = ""

    for i in plainText:
        if (i == " "):
            cipherText += " "
        elif (i.isupper()):
            cipherText += chr((a * (ord(i) - 65) + b) % 26 + 65)
        else:
            cipherText += chr((a * (ord(i) - 97) + b) % 26 + 97)
        print(cipherText)

    return cipherText


def mod_Inv(x, y):
    for i in range(y):
        if (x * i) % y == 1:
            return i


def decryption(cipherText, a, b):
    cipherText = cipherText.rstrip()

    a = mod_Inv(a, 26)

    plainText = ""

    for i in cipherText:
        if (i == " "):
            plainText += " "
        elif (i.isupper()):
            plainText += chr(a * (ord(i) - 65 - b) % 26 + 65)
        else:
            plainText += chr(a * (ord(i) - 97 - b) % 26 + 97)

    return plaintext
```

AtbashC.py

```
def encrypt(string):
    converted_text = ""
    key = {
    'A' : 'Z', 'B' : 'Y', 'C' : 'X', 'D' : 'W', 'E' : 'V', 'F' : 'U', 'G' : 'T', 'H' : 'S', 'I' : 'R',
    'J' : 'Q', 'K' : 'P', 'L' : 'O', 'M' : 'N', 'N' : 'M', 'O' : 'L', 'P' : 'K', 'Q' : 'J', 'R' : 'I',
    'S' : 'H', 'T' : 'G', 'U' : 'F', 'V' : 'E', 'W' : 'D', 'X' : 'C', 'Y' : 'B', 'Z' : 'A', 'a' : 'z',
    'b' : 'y', 'c' : 'x', 'd' : 'w', 'e' : 'v', 'f' : 'u', 'g' : 't', 'h' : 's', 'i' : 'r', 'j' : 'q',
    'k' : 'p', 'l' : 'o', 'm' : 'n', 'n' : 'm', 'o' : 'l', 'p' : 'k', 'q' : 'j', 'r' : 'i', 's' : 'h',
    't' : 'g', 'u' : 'f', 'v' : 'e', 'w' : 'd', 'x' : 'c', 'y' : 'b', 'z' : 'a'}
```

```
    for i in range(len(string)):
        if string[i] in key:
            converted_text += key[string[i]]
        else:
            converted_text += string[i]
    return converted_text
```

AutokeyC.py

```
def encrypt(txt, key):
    if key == "":
        key = "A"
    enc_text = ""
    if len(key) < len(txt):
        key = (key + txt)[: len(txt)]
    elif len(key) > len(txt):
        key = key[: len(txt)]
    for i in range(len(txt)):
        if txt[i].isalpha():
            if txt[i].isupper():
                v = 'A'
            else:
                v = 'a'
            s = (ord(txt[i]) - ord(v) + ord(key[i]) - ord(v)) % 26
            enc_text += chr(s + ord(v))
        else:
            enc_text += txt[i]
    return enc_text

def decrypt(txt, key):
    if key == "":
        key = "A"
    dec_text = ""
    for i in range(len(txt)):
        k = ""
        if txt[i].isalpha():
            if txt[i].isupper():
                v = 'A'
            else:
                v = 'a'
            s = ord(txt[i]) - ord(key[i])
            if s < 0:
                s += 26
            k += chr(s + ord(v))
            key += k
            dec_text += k
        else:
            dec_text += txt[i]
    return dec_text
```

beaufortautokeyC.py

```
def encrypt(txt, key):
    if key == "":
        key = "abc"
    converted_text = ""

    if len(key) > len(txt):
        key = key[: len(txt)]
    elif len(key) < len(txt):
        key += txt
        key = key[:len(txt)]
        # key += txt[:len(key)]
    for i in range(len(txt)):
        if txt[i].isupper():
            v = 'A'
        elif txt[i].islower():
            v = 'a'
        else:
            converted_text += txt[i]
            continue

        s = ord(key[i]) - ord(txt[i])
        if s < 0:
            s += 26

        converted_text += chr(s + ord(v))
    return converted_text

def decrypt(txt, key):
    if key == "":
        key = "abc"
    converted_text = ""

    for i in range(len(txt)):
        b_k = ""
        if txt[i].isupper():
            v = 'A'
        elif txt[i].islower():
            v = 'a'
        else:
            converted_text += txt[i]
            continue

        s = ord(key[i]) - ord(txt[i])
        if s < 0:
            s += 26
        b_k = chr(s + ord(v))
        key += b_k
        converted_text += b_k
    return converted_text
```

beaufortC.py

```
def convert(txt, key):
    spl = ['!','@','#','$','%','^','&','*','(',')','<','~',':',';','<','>','?','/']
    for i in key:
        if i in spl:
            key = key.replace(i,"")
    if key == "":
        key = "abc"
    converted_text = ""

    if len(key) > len(txt):
        key = key[: len(txt)]
    elif len(key) < len(txt):
        key = (key * ((len(txt) // len(key)) + 1))[:len(txt)]

    for i in range(len(txt)):
        if txt[i].isupper():
            v = 'A'
        elif txt[i].islower():
            v = 'a'
        else:
            converted_text += txt[i]
            continue

        s = ord(key[i]) - ord(txt[i])
        if s < 0:
            s += 26

        converted_text += chr(s + ord(v))
    return converted_text
```

caesarC.py

```
def encryption(plainText, key):
    plainText = plainText.rstrip()

    cipherText = ""

    for i in plainText:
        if (i == " "):
            cipherText += " "
        elif (i.isupper()):
            cipherText += chr((ord(i) - 65 + key) % 26 + 65)
        else:
            cipherText += chr((ord(i) - 97 + key) % 26 + 97)

    return cipherText


def decryption(cipherText, key):
    cipherText = cipherText.rstrip()

    plainText = ""
```

```
    for i in cipherText:
        if (i == " "):
            plainText += " "
        elif (i.isupper()):
            plainText += chr((ord(i) - 65 - key) % 26 + 65)
        else:
            plainText += chr((ord(i) - 97 - key) % 26 + 97)

    return plainText


def hacking(cipherText):
    solutions = []

    for i in range(1, 26):
        solutions.append(decryption(cipherText, i))

    return solutions
```

columnerC.py

```
def encrypt(txt, key):
    if key == "":
        key = "key"
    enc_text = ""
    s = ""

    # to make sure elements in key are unique
    for i in key:
        if i not in s: s += i

    key = s
    l = [i for i in key]
    key2 = sorted(list(l))

    if len(key) > len(txt):
        key = key[:len(txt)]

    cols = len(key2)

    # ceil function without using math library
    rows = -(-len(txt) // cols)
    enc_text = ""

    matrix = [[None for c in range(cols)] for r in range(rows)]

    index = 0
    for r in range(rows):
        for c in range(cols):
            if index < len(txt):
                matrix[r][c] = txt[index]
                index += 1

    for i in key2:
```

```
        x = key.index(i)
        enc_text += "".join([row[x] for row in matrix if row[x] is not None])

    return enc_text


def decrypt(txt, key):
    if key == "":
        key = "key"
    s = ""

    for i in key:
        if i not in s: s += i

    key = s
    l = [i for i in key]
    key2 = sorted(list(l))

    if len(key) > len(txt):
        key = key[:len(txt)]

    cols = len(key2)
    rows = -(-len(txt) // cols)
    dec_text = ""
    end = len(txt) % len(key2)

    matrix = [[" " for c in range(cols)] for r in range(rows)]
    diff = len(key) - (len(txt) % len(key))
    if diff == len(key):
        diff = 0

    cl = -1
    while diff != 0:
        matrix[-1][cl] = None
        diff -= 1
        cl -= 1

    indx = 0
    for i in key2:
        x = key.index(i)
        for r in range(rows):
            if matrix[r][x] is not None and indx < len(txt):
                matrix[r][x] = txt[indx]
                indx += 1

    for r in range(rows):
        for c in range(cols):
            if matrix[r][c] is not None:
                dec_text += matrix[r][c]
    return dec_text
```

HomophonicC.py

```
import random

key_table = {'A': 'D9', 'B': 'X', 'C': 'S', 'D': 'F', 'E': 'Z721', 'F': 'E', 'G': 'H', 'H': 'C', 'I': 'V3',
             'J': 'I', 'K': 'T', 'L': 'P', 'M': 'G', 'N': 'A5', 'O': 'Q0', 'P': 'L', 'Q': 'K', 'R': 'J', 'S': 'R4',
             'T': '6U',
             'U': 'O', 'V': 'W', 'W': 'M', 'X': 'Y', 'Y': 'B', 'Z': 'N', 'a': 'd(', 'b': 'x', 'c': 's', 'd': 'f',
             'e': 'z&@!',
             'f': 'e', 'g': 'h', 'h': 'c', 'i': 'v#', 'j': 'i', 'k': 't', 'l': 'p', 'm': 'g', 'n': 'a%', 'o': 'q)',
             'p': 'l',
             'q': 'k', 'r': 'j', 's': 'r$', 't': '^u', 'u': 'o', 'v': 'w', 'w': 'm', 'x': 'y', 'y': 'b', 'z': 'n'}


def encryption(txt):
    enc_text = ""
    for item in txt:
        if item in key_table:
            enc_text += random.choice(key_table[item])
        else:
            enc_text += item
    return enc_text


def decryption(txt):
    dec_text = ""
    flag = 1
    words = txt.split(" ")

    lst = []
    for item in words:
        lst += item + " "

    for letter in lst:
        if letter == " ":
            dec_text += " "
        else:
            for item in key_table:
                flag = 1
                if letter in key_table[item]:
                    dec_text += item
                    flag = 0
                    break
            if flag == 1:
                dec_text += letter
    return dec_text
```

Input and Output Screen

Testing and Validation Check

Testing and Validation Check

Validation is nothing but the security measures taken at the time of the execution of any program. It is necessary for the analyst to take the validation in their project as it provides more accuracy and systematic flow to project. Validation not only stops input of the false data but also provides the information in the form of message to the user clearly warn the user to input correct data type. Hence it plays an important role of a guide during input of data.

Validation put it controls over the data in both character as well as integer data type. Whenever wrong data or invalid data is stored by the user it frees the message immediately.

Validation input transaction:

Validation input data is largely done through software which is the programmer's responsibility but it is important that system analyst must know that common program might invalidate a transaction. Business committed to quality will include validation checks as a part of their routine software.

Following are the situations where there's need to validate input data

1) Submitting the wrong data to system.

2) Submitted the data by an unauthorized person.

3) Asking the system to perform an unacceptable function.

Validation input data:

It is essential that the input themselves along with the transaction requested are valid. Several texts can be incorporated into software to ensure the validity. We consider many possible ways to validate input and they are as follows:

- Validation is done for the empty fields that if at all any field remains empty then application will prompt to enter the data in all the fields.

- Test is done for the name field that only character(data type) will be allowed to enter.
- To input e-mail id in the textbox it is compulsory to enter the valid email address like an email address should contain @gmail.com into its mail string.
- Password can be of maximum 45 characters. It can be digits, letters or any special characters.
- Password entered in the form is not seen till the show password checkbox is not checked.
- Username should be entered to go on the forgot password form.
- Test is made with the data available in the database to get login with the application or to cancel the admission or to edit the profile of the student.
- Only numbers are allowed when entering the pin code.

Implementation Evaluation and Maintenance

Implementation, evaluation and Maintenance

Implementation phase is mainly concerned with the user training, site, and preparation and file conversion. It also involves final testing of the system.

During implementation the component built during development are put into optional use.

Following are the points should be considered while doing implementation of the application:

- Testing, debugging and documentation program.
- Converting data from old to new system.
- Giving training to the user about how to operate the system.
- Developing operating procedures for the computer operating staff.
- Establishing a maintenance procedure to repair and enhance system.
- Completing Documentation
- Operating system on the user location and solving all the issues
- occurred while operation.

Evaluation

After the implementation phase, another stage in project development is evaluation. After keeping the project in the working condition for some time, all the errors that are shown in the computer program should be removed. The programmer needs to correct them so that same errors should not be repeated. We should also get the feedback from the user which are using it and ask them whether, it is user friendly or not. After evaluating the program and satisfying the needs of the user the program is maintained fully to give the same functionalities for what it was intended to be. This stage should be implemented so as to regular check-up the errors with error handling techniques. This stage is the updating and correcting of the program to account for changing condition or field experience. Proper testing and Documentation significantly reduce the frequency and extent of the required maintenance.

Maintenance

Maintenance is very crucial for success of any application; proper maintenance of the application makes it smooth working application. Maintenance is done basically, for two reasons i.e., to correct software errors which occurs after the testing and implementation of the application when one user it and other reason is to enhance the software capabilities in response to changing organizational needs. User often requires additional feature after he/she uses the application and becomes familiar with it. Some of the large companies gives AMC (Annual Maintenance Contract) to other companies for regular maintenance of the software/application. The cost of the maintenance increases the cost of the application/software. At a point of time, it becomes feasible to perform the tasks related to the maintenance of the software. Maintenance phase always occurs after the implementation of the application is done. It corrects all the previously undetected errors of the application and helps to do update in the application which is required by the user. Maintenance is one of the stages in the SDLC (System Development Life Cycle). It is basically done for the estimation, controlling, and making modification to the implemented system.

Future scope of the project

Future scope of the project

In future there is a plethora of actions a user can perform on our encryption software.

- More than Text: Currently focusing on only encrypting a piece of text, but in future, the user will be able to encrypt audio files, images, video files by converting them into a text-based format which is only readable by our application on the basis of what key the user has used.
- Creating own algorithm: The best future update is when the users are provided with the option of creating their own algorithm which will be used to encrypt the file. Developers will especially like this option as they will be able to create a key if they wish and then they will know that their own encryption is the best as no one except themselves will be able to decode it. This also provides an added layer of security.
- Speech to Text and vice versa: The users will be given the option of conversion of their own speech directly to text as it allows easy functioning and saves time for the users. This also allows people with disabilities to either listen to what they've input or easily input their own data without having the need to type.
- Direct Encryption of text files: As of now, the user has to type in their text but later on they will be able to directly choose a file with textual extensions such as .txt, .doc, .docx etc.
- Option to convert to PDF or other formats: Later the user will be able to select a text file and either edit it to add more information or to directly convert the encrypted file to multiple formats.
- Online Encrypted Mail without using email applications: Connectivity to internet allows the user to directly send their encrypted data via mail to

another user which they've added on their application as a known person. This allows even more security in the transit of data.

- Connection to internet shall allow access of data everywhere: Connectivity of internet shall allow access to a user to access their key or their saved file on cloud from anywhere they want globally only if they have the app installed as making it into a website shall increase load to save it on the cloud and also make it susceptible to online threats.

Conclusion

Conclusion

The program created by us has a huge potential to develop into a software which is easily accessible to users and which they can use to encode multiple files and directly convert them into a PDF file or a word file or others.

Also, later on we will develop this software to encrypt more than just text such as encryption of images and converting them into encrypted text or encrypting an audio file so that no one can figure out the use of that text file or decode it into an audio file without a proper key.

Ours might not be the most visually pleasing , but it surely is the one that required great amount of logic and coding and one of the most important ones.

Bibliography:

- Cracking Codes with Python – Al Sweigart
- Hacking Secret Ciphers with Python – Al Sweigart

References:

https://youtu.be/S-w24LtBub8

https://towardsdatascience.com/encrypt-and-decrypt-files-using-python-python-programming-pyshark-a67774bbf9f4

https://devqa.io/encrypt-decrypt-data-python/

https://www.thesslstore.com/blog/types-of-encryption-encryption-algorithms-how-to-choose-the-right-one/

https://www.keyfactor.com/resources/types-of-encryption-algorithmns/