IMPROVISING IMAP PROTOCOL BY INTEGRATING SPAM DETECTION MODEL AND IMPLEMENTING SECURE COMMUNICATION USING RSA CRYPTOGRAPHY

Report submitted to the SASTRA Deemed to be University as the requirement for the course

CSE302: COMPUTER NETWORKS

Submitted by

INTI DHIRAJ
124157027, B. Tech – CSE(CSBC)

December 2022

SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA – 613 401

SCHOOL OF COMPUTING

THANJAVUR – 613 401

Bonafide Certificate

This is to certify that the report titled "**Improvising IMAP protocol by integrating spam detection model and implementing secure communication using RSA cryptography**" submitted as a requirement for the course, **CSE302: COMPUTER NETWORKS** for B.Tech. is a bonafide record of the work done by **Shri. Inti Dhiraj (Reg. No.124157027, CSE(CSBC)**) during the academic year 2022-23, in the School of Computing.

Project Based Work *Viva voic*e held on ________________

Examiner 1 **Examiner 2**

List of figures

Figure number	**Description**	**Page Number**
3.1	Client Server Mode	3
3.2	SMTP server	4
3.3	Extraction by IMAP server	5
3.4	RSA generating public key	6
3.5	RSA generating private key	6
3.6	Accuracy of the model	8
3.7	The work flow	8
5.1 – 5.10	Output snapshots	20 - 23

Abbreviations

ESMTP	Extended Simple Mail Transfer Protocol
IMAP	Internet Message Access Protocol
IP	Internet Protocol
ML	Machine Learning
MIME	Multipurpose Internet Mail Extensions
MTA	Mail Transfer Agent
RSA	Rivest Shamir Adleman
SMTP	Simple Mail Transfer Protocol
TCP	Transmission Control Protocol

Abstract

Nowadays there are increasing problems of ransomware, spam mails. Hence this project develops an application in which there should be a secure communication amongst the clients of a server, although a spam is received the filter integrated with the protocol will ensure that it alerts the user via the application/platform used and will delete the mail with user consent, hence we can save unnecessarily occupied memory by spams in the server as well, as nowadays data storage is a concern as well. To work towards building the solution we must first make use of globally available IMAP servers from application using simple python programs, similarly for SMTP. Also, we should implement RSA modules for secure communication over various servers. Finally build a spam filter using an appropriate machine learning algorithm, logistic regression model is used here. This work concentrates on the current security standards adopted in practice by providers to safeguard the communications among their SMTP servers.

KEY WORDS: IMAP, SMTP, RSA, Machine Learning

Table of Contents

Title	Page No.
Bonafide Certificate	ii
List of figures	iii
Abbreviations	iv
Abstract	v
1. INTRODUCTION	1
2. FEATURES IMPLEMENTED	2
3. RELATED WORK	3
4. SOURCE CODE	9
5. SNAPSHOTS .	20
6. CONCLUSION & FUTURE WORK	24
7. REFERENCES	25

1. INTRODUCTION

It's no secret that one of the most valuable commodities in the modern world is data. A person's information can be utilized in a variety of ways, from helping nefarious conspiracies to sabotage important activities to supporting seemingly innocent targeted marketing on social media. There is a need for secure communication methods as a result, particularly for emails that are sent from one party to another using a client-server architecture.

Client-server architecture is no longer an unfamiliar concept in computer networks these days. In fact, the internet up to this date has many client server applications readily available anytime. Some examples of client-server applications are web access, network time and windows login. In client- server model, each computer terminal or process on the network could be a client or a server. A client is a program or a computer terminal that allows users to access and view its interfaces. Every client connects and communicates with each other through server. In this process of communication, client serves as the one who initiates the communication wherein the server is the one who waits passively to respond to the client's request.

Hence, developed simple networking programs using python programming language and a spam filter for detecting the spam messages that are incoming, and deleting the spam messages regularly which are not deleted by users on a regular basis, hence saving the memory. Also, secure communication is implemented using RSA cryptosystem, which improves confidentiality of communication between sender and receiver over third party servers.

1.1. MERITS:

1. SMTP ensures secure communication when using port number 465.
2. IMAP ensures secure communication over port number 993.
3. The spam filter will segregate the spam mails, alert the user about the same.
4. Confidentiality of communication is maintained

1.2. DEMERITS:

1. Although the port of SMTP and IMAP are secure, one may pretend to be the sender and the receiver may be deceived.
2. The filter has accuracy of 96.70% i.e., there is a chance of 3.30% of receiving spam

2. FEATURES IMPLEMENTED

2.1. Detecting spam and taking action:

An application is developed, which can send a mail, from a given mail, without actually logging into the SMTP server of the email user, i.e., the client can log in to his/her email account with this application remotely and type in the mail they wish to send. Similarly, on the receiver side, again the client side, will login to their email account using the IMAP server of their email, and can extract the email and read the same. The application will tell whether it is a spam or not, if it is a spam then it will ask the user, whether the mail should be deleted to free up the space.

➢ Functionalities performed by this application are

- **Sending mail:** This feature enables a user to send a mail to all other clients connected to the server. (SMTP protocol)
- **Receiving mail:** This feature enables a user to receive a mail from a particular client connected to the server. (IMAP protocol)
- **Spam filter:** This feature makes sure that it detects the spam mail, and seeks permission to delete redundant spam mails. (ML model)

2.2. Secure communication:

This application also does implement the feature of secure communication by means of RSA cryptosystem, the public key has to be shared with authenticated user and the private key is receiver initiated hence ensures the authentication and confidentiality.

➢ Functionalities performed by this application are

- **Encryption:** This function encrypts the given data with a public key which is the receiver initiated key.
- **Decryption:** This function decrypts the given data with a private key which is the receiver initiated key.

3. CONCEPTS/RELATED WORK

3.1. Client-Server Mode:

The use of client server model is based on its request – reply protocol. In this model, the client directly sends his request to the server asking for an appropriate service, and later server does the work and returns immediately the data or error code as a response back to the client. The server responds back to client with the service requested by the client.

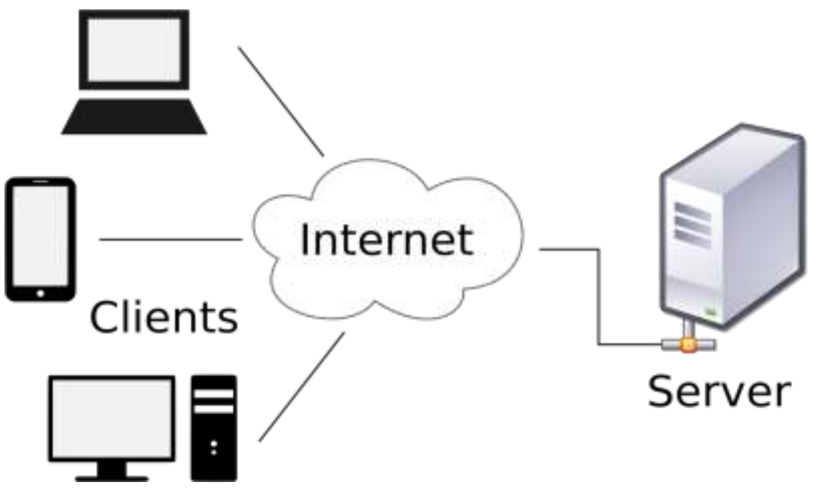


Fig.3.1. Client-Server Mode

3.2. SMTP protocol:

An SMTP server is a mail server that forwards emails from a sender to one or more recipients. The main function of the mail server is to prevent mail from being sent to people who are not authorized to do so. The protocol extension ESMTP with SMTPAuth is needed to enable this modern mail server. The senders outgoing mail server and the recipients incoming mail server are both involved in the email transmission process through the relays SMTP server.

After the sender has sent their email the webmail application or program used to send the email known as the SMTP client or Mail User Agent converts the message into a body before loading it on to the outgoing mail server. The software used for sending and receiving emails is represented by the MTA. The email is checked by the MTA and then saved. The MTAs role is sometimes preceded by a Mail Submission Agent with the function of checking the validity of emails. The MTA checks the recipient's email address.

SMTP is an application layer protocol. The client who wishes to send the mail and opens a TCP connection to the SMTP server, here in this case it is G-mail SMTP server and then sends the mail across the connection. The G-mail SMTP server is always listening. As soon as a TCP connection request arrives at the server port, it initiates the TCP connection and the client starts sending the mail using SMTP and MIME. Here to demonstrate the working of SMTP protocol, the communication between remote host and the SMTP server is also shown in the output.

- ❖ **Advantages**

  - ➢ All you need to do to make it function is enter your credentials.
  - ➢ If email delivery is unsuccessful, a detailed explanation will be included in the message.
  - ➢ Starting to use mail for your transactional emails is quite simple. All you need to do to get started is swap ceremonial. unlike APIs, where programming is necessary.

- ❖ **Disadvantages**

  - ➢ Some firewalls have the ability to block the SMTP port.
  - ➢ The SMTP security issue is becoming worse.
  - ➢ Without transforming them into text files, binary files cannot be sent through SMTP. To transmit mail in a different format, use MIME.

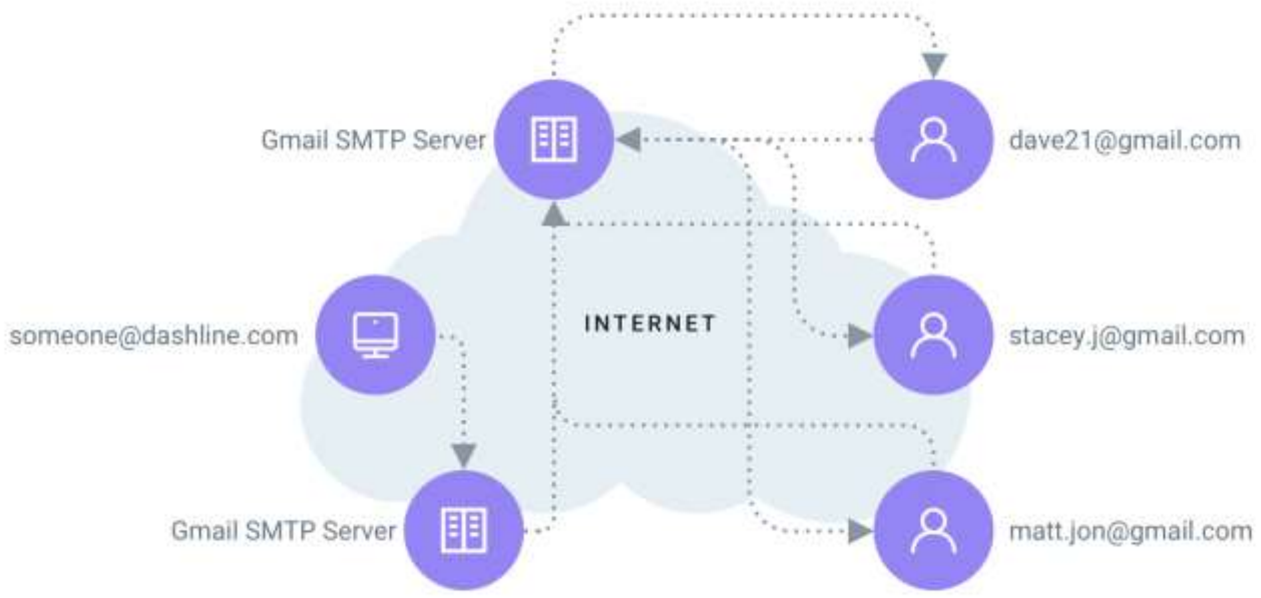


Fig 3.2 SMTP server

3.3. IMAP protocol:

IMAP is an application layer protocol. The client on the receiver side will be logging into his/her email account and will be fetching top 'N' messages in his/her INBOX folder. Each mail's sender address, subject and content will display and if any attachments found, they will be saved as a file under a folder with subject as its name. As an incoming email protocol, IMAP functions as the intermediary between the email server and email client. When users read an email using IMAP, they read them off the server. They don't actually download or store the email on their local device. This means that the email is not tied to a particular device, and users can access it from any location in the world using different devices.

Interactive mail access protocol, Internet mail access protocol, and interim mail access protocol are other names for this technique. Despite its widespread use, IMAP is no longer as crucial given how much email is being sent via web-based interfaces like Gmail, Hotmail, Yahoo Mail, etc.

It is a method of getting access to email messages that are stored on a server without having to download them to the user's local hard disc. This is the primary distinction between IMAP and POP3, a well-known email protocol.

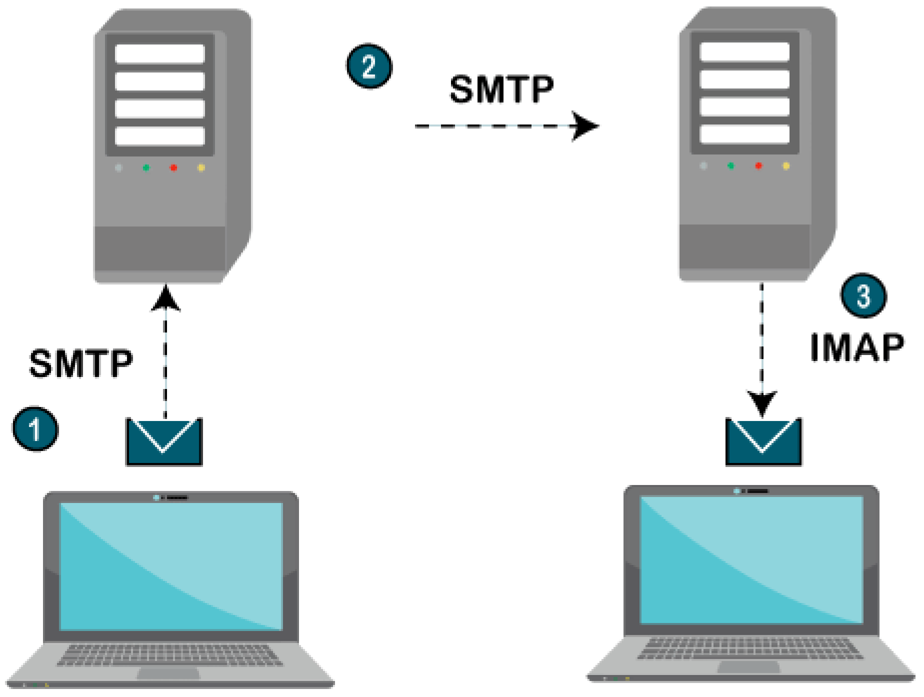


Fig 3.3 Extraction by IMAP server

❖ **Advantages**

- It provides synchronization between all of the user-maintained sessions.
- Due to the fact that emails are stored solely on the IMAP server, it offers security over POP3 protocol.
- All of the contents are accessible to users remotely.
- Because it is synced by a centralized server, it allows for simple migration across the devices.
- No physical space needs to be set aside to store contents.

❖ **Disadvantages**

- IMAP maintenance is difficult.
- The user's emails are only accessible when an internet connection is present.
- The loading of messages takes longer.
- IMAP is not supported by all emails, which makes management challenging.
- Several browser-based solutions aren't available because IMAP isn't supported.

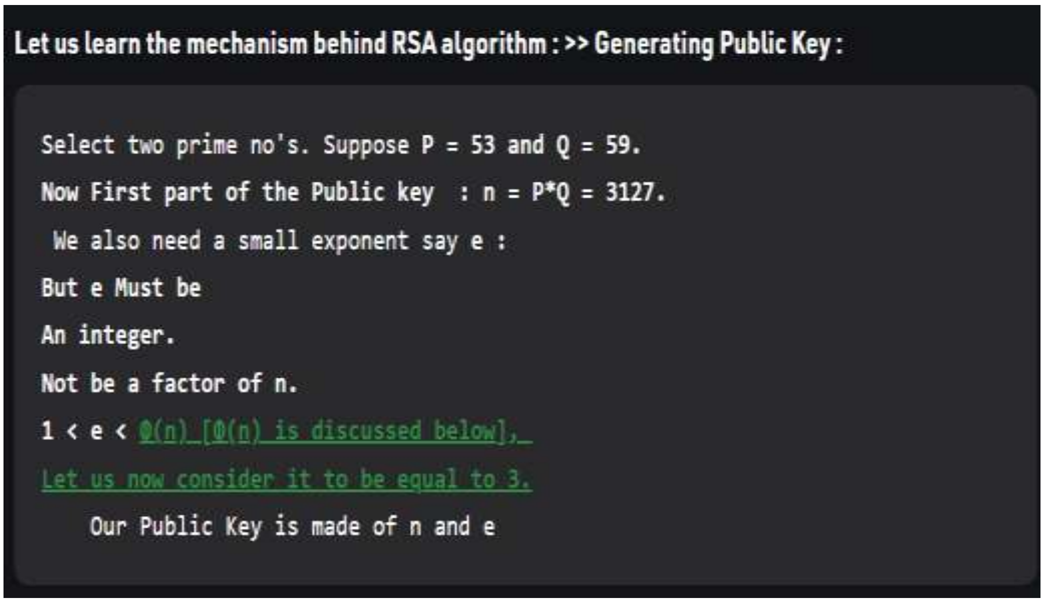
3.4. RSA cryptosystem:

RSA cryptosystem is one of the classical cryptography algorithms in which the keys are receiver generated key pairs, the public key is given to the authoritative sender and the private key is kept with the receiver. The assumption made while designing RSA system is it is difficult to factorize a large number which is in fact a product of two prime numbers. The public key consists of two numbers where one number is a multiplication of two large prime numbers. Private key is also derived from those two primes numbers. So, suppose the large number gets factorized, the private key will be unearthed. Therefore, encryption strength totally lies on the key size, solution to this can be increasing the size of the key, or the prime numbers contributing to form the large number. It is a receiver-initiated algorithm hence ensures confidentiality.

Step 1: Alice encrypts the message using Bob's public key in step one.

Step 2: Bob receives the encrypted message

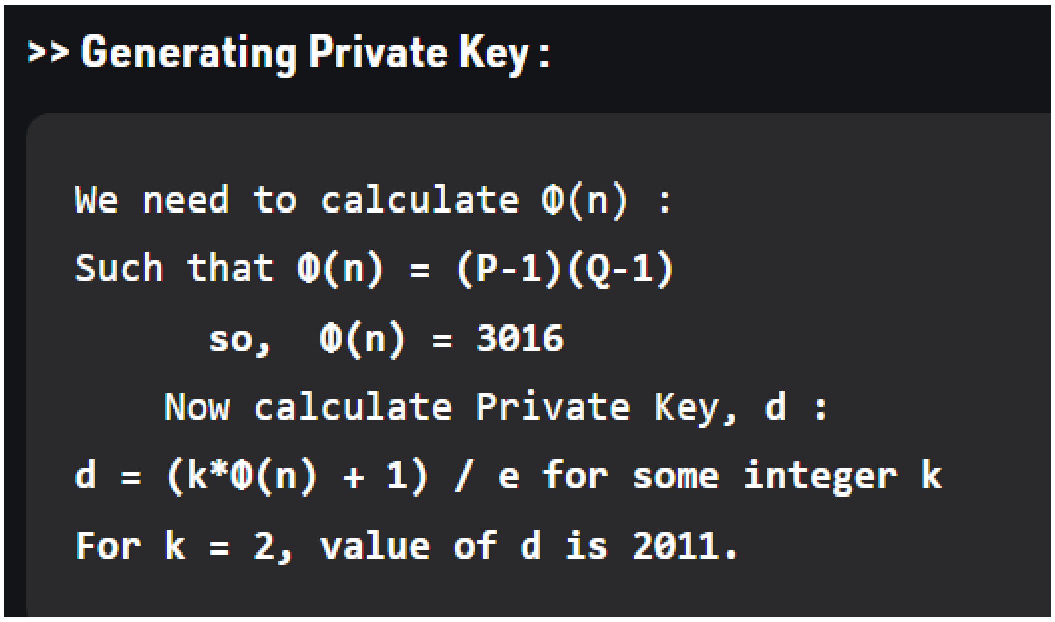
Step 3: Bob decrypts the message using his private key



Let us learn the mechanism behind RSA algorithm : >> Generating Public Key :

```
Select two prime no's. Suppose P = 53 and Q = 59.
Now First part of the Public key  : n = P*Q = 3127.
 We also need a small exponent say e :
But e Must be
An integer.
Not be a factor of n.
1 < e < Φ(n) [Φ(n) is discussed below],
Let us now consider it to be equal to 3.
    Our Public Key is made of n and e
```

Fig 3.4 RSA generating public key



>> Generating Private Key :

```
We need to calculate Φ(n) :
Such that Φ(n) = (P-1)(Q-1)
      so,  Φ(n) = 3016
    Now calculate Private Key, d :
d = (k*Φ(n) + 1) / e for some integer k
For k = 2, value of d is 2011.
```

Fig 3.5 RSA generating private key

❖ **Advantages**

- No Key Sharing: Since RSA encryption uses the recipient's public key, receiving messages from others does not need the sharing of any secret keys.
- A receiver cannot intercept a message because they lack the correct private key to decrypt the data because the key pairs are tied to one another.
- Faster encryption: Compared to the DSA method, encryption is completed more quickly.
- Cannot Change Data: Since tampering with the data will change how the keys are used, the data will remain secure while in transit. Additionally, the information cannot be decrypted using the private key, alerting the recipient to manipulation.

❖ **Disadvantages**

- Because RSA only employs asymmetric encryption and complete encryption requires both symmetric and asymmetric encryption, it might occasionally fail.
- The high volume of participants causes the slow data transfer rate.
- Sometimes a third party is needed to confirm the validity of public keys.
- Decryption requires intensive processing on the receiver's end.

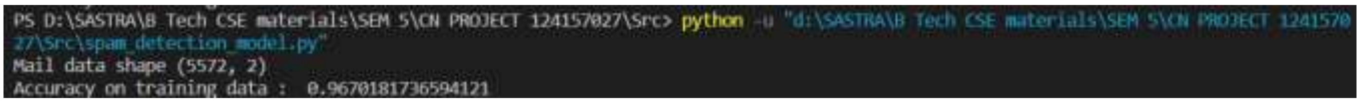
3.5. The Spam Filter:

Basically, one of the aims of this project is to build an accurate spam filter to filter out the spams, hence a Machine Learning based spam filter is integrated with the application on the receiver side to detect the same. This is built with an accuracy of 96.70%, which is nothing but its accuracy score. Here we have used Logistic Regression, as it is a multivariable/ multi feature dependent output prediction. It is a supervised learning model. The model has been trained with over 5000 mails. The model uses a count vectorizer for words occurring in the training dataset and writes it as a binary file, also saves the model as a binary file.

A program known as a spam filter is used to identify unsolicited, undesired, and virus-infected emails and stop them from reaching a user's mailbox. A spam filter, like other filtering systems, searches for certain standards on which to base its decisions.

Email spam filtering systems are used by Internet service providers (ISPs), enterprises, and free online email services to reduce the danger of spam distribution. For instance, one of the first and most basic spam filtering systems, like the one used by Microsoft's Hotmail, was configured to look for specific terms in message subject lines. When the filter detected one of the predefined terms, the email was not added to the user's inbox.

This approach is not particularly efficient and frequently leaves out completely acceptable messages, known as false positives, while allowing true spam messages to pass. Using suspicious word patterns or word frequency, more advanced systems, including Bayesian filters and other heuristic filters, can identify spam messages. They accomplish this by gaining insight into the user's preferences from the emails flagged as spam. When new emails are sent that are intended for the user's inbox, the spam program creates rules and applies them. For instance, the Bayesian filter detects a trend when users flag emails as spam and automatically adds all subsequent emails from that.



```
PS D:\SASTRA\B Tech CSE materials\SEM 5\CN PROJECT 124157027\Src> python -u "d:\SASTRA\B Tech CSE materials\SEM 5\CN PROJECT 1241570
27\Src\spam_detection_model.py"
Mail data shape (5572, 2)
Accuracy on training data :  0.9670181736594121
```

Fig 3.6 Accuracy of the model

3.6. The work flow:

Fig 3.7 explains about how this application works on the clients’ side, on the sender side the message is encrypted and sent using SMTP and MIME protocols to the their host SMTP server and then it is sent to the destination host IMAP server from which the receiver side’s application will retrieve the email and decrypt using the appropriate key, which is receiver initiated, hence decrypts with his private key, the decrypted message is fed to spam filter to know if it is a spam or not, if it is a spam then it prompts user to click yes or no on the GUI, else it just shows the email body and result as ham.

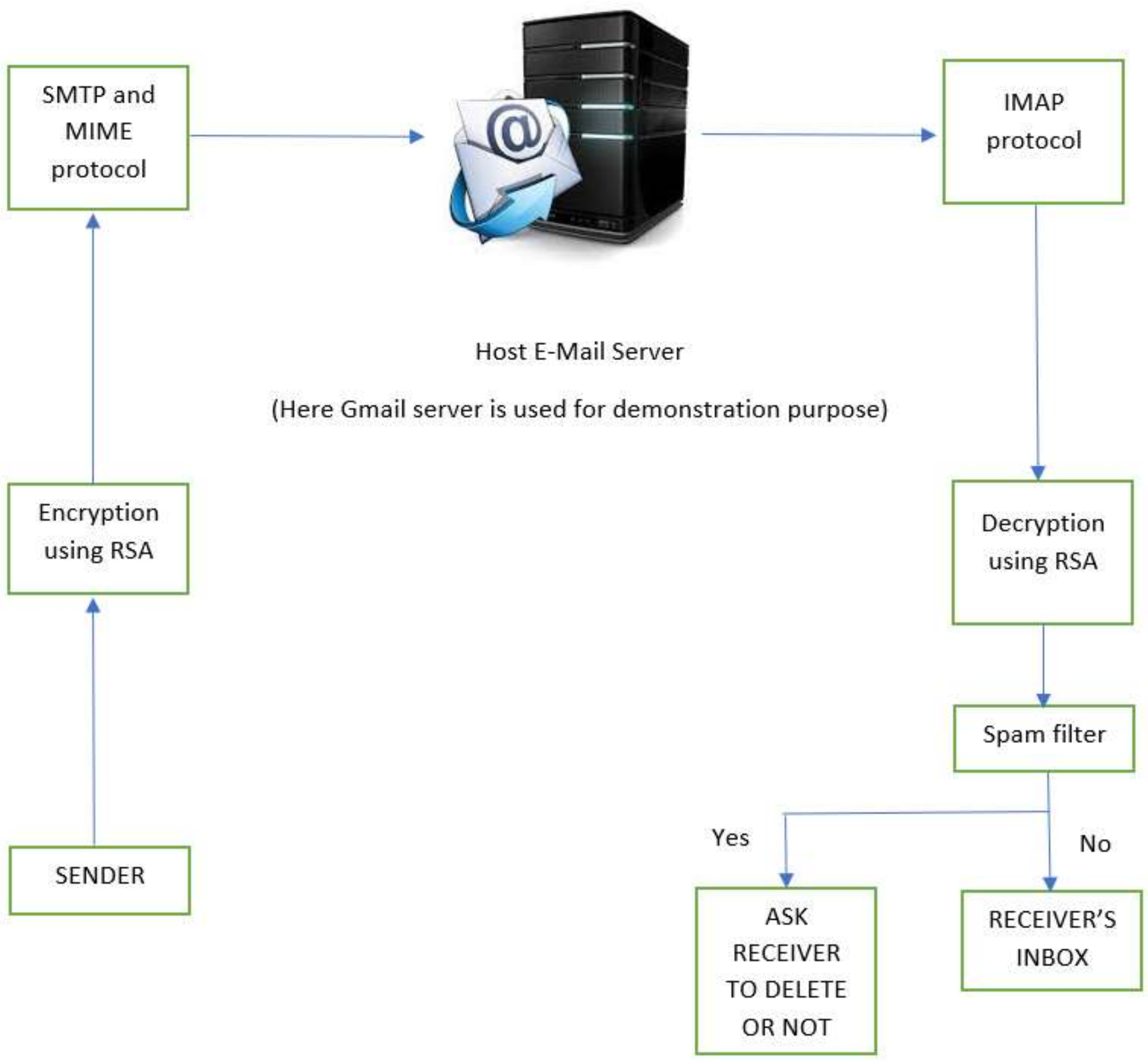


Fig 3.7 The work flow

4. SOURCE CODE

4.1. SMTP Code:

```
from email.mime.text import MIMEText
from email.mime.image import MIMEImage
from email.mime.application import MIMEApplication
from email.mime.multipart import MIMEMultipart
import smtplib
import rsa_impl
import gui_input

with open('keys.txt','r') as f:
    d,e,n = f.readline().split()
d = int(d)
e = int(e)
n = int(n)
private, public = (d,n),(e,n)


smtp = smtplib.SMTP('smtp.gmail.com', 587)
smtp.ehlo()
smtp.starttls()
smtp.login('124157027@sastra.ac.in', 'hlgxyjzcxrcccrtr')

msg = MIMEMultipart()
msg['Subject'] = "Machine generated mail"

#text = "07732584351 - Rodger Burns - MSG = We tried to call you re your reply to our sms for a free nokia mobile + free camcorder. Please call now 08000930705 for delivery tomorrow"
#text = "As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your callertune for all Callers. Press *9 to copy your friends Callertune"
text = open('mail_to_be_sent.txt','r').read()
text = rsa_impl.encrypt(public,text)
text = [str(i) for i in text]
#print(type(text))
text = " ".join(text)
msg.attach(MIMEText(text))
#print(text)
smtp.set_debuglevel(True)

#to = 'intidhiraj3103@gmail.com'
to = "124157027@sastra.ac.in"

smtp.sendmail(from_addr="124157027@sastra.ac.in",to_addrs=to, msg=msg.as_string())
smtp.quit()
```

4.2. IMAP code:

```
import imaplib
import email
from email.header import decode_header
import webbrowser
import os
import rsa_impl
import pickle

import tkinter as tk
import tkinter.font as tkFont

class App:
    def __init__(self, root):
        #setting title
        root.title("undefined")
        #setting window size
        width=600
        height=500
        screenwidth = root.winfo_screenwidth()
        screenheight = root.winfo_screenheight()
        alignstr = '%dx%d+%d+%d' % (width, height, (screenwidth - width) / 2, (screenheight - height)
/ 2)
        root.geometry(alignstr)
        root.resizable(width=False, height=False)

        GLabel_183=tk.Label(root)
        ft = tkFont.Font(family='Times',size=10)
        GLabel_183["font"] = ft
        GLabel_183["fg"] = "#333333"
        GLabel_183["justify"] = "center"
        GLabel_183["text"] = "Mail received:"
        GLabel_183.place(x=90,y=160,width=100,height=42)

        GLabel_120=tk.Label(root)
        ft = tkFont.Font(family='Times',size=10)
        GLabel_120["font"] = ft
        GLabel_120["fg"] = "#333333"
        GLabel_120["justify"] = "center"
        GLabel_120["text"] = "Developed by Inti Dhiraj"
        GLabel_120.place(x=160,y=40,width=214,height=30)

        GLabel_554=tk.Label(root)
        ft = tkFont.Font(family='Times',size=10)
        GLabel_554["font"] = ft
        GLabel_554["fg"] = "#333333"
        GLabel_554["justify"] = "center"
        GLabel_554["text"] = "Result: "
        GLabel_554.place(x=100,y=310,width=70,height=25)

    def GButton_63_command(self):

        status, messages = imap.search(None, 'FROM "124157027@sastra.ac.in"')
        messages = messages[0].split(b' ')
        for mail in messages:
            _, msg = imap.fetch(mail, "(RFC822)")
            # you can delete the for loop for performance if you have a long list of emails
            # because it is only for printing the SUBJECT of target email to delete
            for response in msg:
                if isinstance(response, tuple):
                    msg = email.message_from_bytes(response[1])
                    # decode the email subject
                    subject = decode_header(msg["Subject"])[0][0]
                    if isinstance(subject, bytes):
                        # if it's a bytes type, decode to str
                        subject = subject.decode()
                    print("Deleting", subject)
            # mark the mail as deleted
            imap.store(mail, "+FLAGS", "\\Deleted")
        # permanently remove mails that are marked as deleted
        # from the selected mailbox (in this case, INBOX)
        imap.expunge()
        root.destroy()


    def GButton_58_command(self):
        root.destroy()

with open('keys.txt','r') as f:
    d,e,n = f.read().split()

d = int(d)
e = int(e)
n = int(n)

private, public = (d,n),(e,n)



user = '124157027@sastra.ac.in'
password = 'npxijznbuwvdmpwk'
host = 'imap.gmail.com'

# Connect securely with SSL
imap = imaplib.IMAP4_SSL(host)

## Login to remote server
imap.login(user, password)
#imap.debug = 5
status, messages = imap.select("INBOX")
# number of top emails to fetch
N = 1
# total number of emails
messages = int(messages[0])
```

```
def clean(text):
    # clean text for creating a folder
    return "".join(c if c.isalnum() else "_" for c in text)

for i in range(messages, messages-N, -1):
    # fetch the email message by ID
    res, msg = imap.fetch(str(i), "(RFC822)")
    for response in msg:
        if isinstance(response, tuple):
            # parse a bytes email into a message object
            msg = email.message_from_bytes(response[1])
            # decode the email subject
            subject, encoding = decode_header(msg["Subject"])[0]
            if isinstance(subject, bytes):
                # if it's a bytes, decode to str
                subject = subject.decode(encoding)
            # decode email sender
            From, encoding = decode_header(msg.get("From"))[0]
            if isinstance(From, bytes):
                From = From.decode(encoding)
            print("Subject:", subject)
            print("From:", From)
            # if the email message is multipart
            if msg.is_multipart():
                # iterate over email parts
                for part in msg.walk():
                    # extract content type of email
                    content_type = part.get_content_type()
                    content_disposition = str(part.get("Content-Disposition"))
                    try:
                        # get the email body
                        body = part.get_payload(decode=True).decode()
                    except:
                        pass
                    if content_type == "text/plain" and "attachment" not in content_disposition:
                        # print text/plain emails and skip attachments
                        #print(body)
                        print()
                    elif "attachment" in content_disposition:
                        # download attachment
                        filename = part.get_filename()
                        if filename:
                            folder_name = clean(subject)
                            if not os.path.isdir(folder_name):
                                # make a folder for this email (named after the subject)
                                os.mkdir(folder_name)
                            filepath = os.path.join(folder_name, filename)
                            # download attachment and save it
                            open(filepath, "wb").write(part.get_payload(decode=True))
            else:
                # extract content type of email
                content_type = msg.get_content_type()
                # get the email body
                body = msg.get_payload(decode=True).decode()
                if content_type == "text/plain":
                    # print only text email parts
                    print(body)
            #print(content_type)
            #print(body)
            if content_type == "text/html":
                # if it's HTML, create a new HTML file and open it in browser
                folder_name = clean(subject)
                if not os.path.isdir(folder_name):
                    # make a folder for this email (named after the subject)
                    os.mkdir(folder_name)
                filename = "index.html"
                filepath = os.path.join(folder_name, filename)
                # write the file
                open(filepath, "w").write(body)
                # open in the default browser
                webbrowser.open(filepath)
            print("="*100)

#load model
model = pickle.load(open('model.pkl', 'rb'))
body = body.split()
body = [int(i) for i in body]
body = rsa_impl.decrypt(private,body)
print(body)
input_mail = [body]

# convert text to feature vectors and load vectorizer
feature_extraction = pickle.load(open('vectorizer.pickle', 'rb'))
input_data_features = feature_extraction.transform(input_mail)

# making prediction
prediction = model.predict(input_data_features)
print("It is a",end=' ')

if prediction[0] == 1:
    print('Ham mail')
    spam_ham = 'Ham'
else:
    print('Spam mail')
    spam_ham = 'Spam'

print("="*100)




root = tk.Tk()
```

```
app = App(root)
GMessage_468=tk.Message(root)
ft = tkFont.Font(family='Times',size=10)
GMessage_468["font"] = ft
GMessage_468["fg"] = "#333333"
GMessage_468["justify"] = "center"
GMessage_468["text"] = body
GMessage_468.place(x=230,y=90,width=270,height=199)

GLabel_932=tk.Label(root)
ft = tkFont.Font(family='Times',size=10)
GLabel_932["font"] = ft
GLabel_932["fg"] = "#333333"
GLabel_932["justify"] = "center"
GLabel_932["text"] = spam_ham
GLabel_932.place(x=320,y=310,width=70,height=25)

if spam_ham=='Spam':
    GLabel_831=tk.Label(root)
    ft = tkFont.Font(family='Times',size=10)
    GLabel_831["font"] = ft
    GLabel_831["fg"] = "#333333"
    GLabel_831["justify"] = "center"
    GLabel_831["text"] = "Delete the mail?(yes/no)"
    GLabel_831.place(x=90,y=400,width=130,height=41)

    GButton_63=tk.Button(root)
    GButton_63["bg"] = "#f0f0f0"
    ft = tkFont.Font(family='Times',size=10)
    GButton_63["font"] = ft
    GButton_63["fg"] = "#000000"
    GButton_63["justify"] = "center"
    GButton_63["text"] = "Yes"
    GButton_63.place(x=290,y=410,width=70,height=25)
    GButton_63["command"] = app.GButton_63_command

    GButton_58=tk.Button(root)
    GButton_58["bg"] = "#f0f0f0"
    ft = tkFont.Font(family='Times',size=10)
    GButton_58["font"] = ft
    GButton_58["fg"] = "#000000"
    GButton_58["justify"] = "center"
    GButton_58["text"] = "No"
    GButton_58.place(x=400,y=410,width=70,height=25)
    GButton_58["command"] = app.GButton_58_command
root.mainloop()


# close the connection and logout
imap.close()
imap.logout()
```

4.3. RSA implementation:

```
import random
import sympy

def gcd(a, b):
    while b != 0:
        a, b = b, a % b
    return a

def multiplicative_inverse(e, phi):
    d = 0
    x1 = 0
    x2 = 1
    y1 = 1
    temp_phi = phi

    while e > 0:
        temp1 = temp_phi//e
        temp2 = temp_phi - temp1 * e
        temp_phi = e
        e = temp2

        x = x2- temp1* x1
        y = d - temp1 * y1

        x2 = x1
        x1 = x
        d = y1
        y1 = y

    if temp_phi == 1:
        return d + phi

def generate_keypair(p, q):
    n = p*q
    phi = ((p-1)*(q-1))

    e = random.randrange(1, phi)

    g = gcd(e, phi)
    while g != 1:
        e = random.randrange(1, phi)
        g = gcd(e, phi)

    d = multiplicative_inverse(e, phi)

    return ((e, n), (d, n))
```

```
def encrypt(pk, plaintext):
    key, n = pk
    cipher = [(ord(char) ** key) % n for char in plaintext]
    return cipher


def decrypt(pk, ciphertext):
    key, n = pk
    plain = [chr((char ** key) % n) for char in ciphertext]
    return ''.join(plain)


""" p = sympy.randprime(1, 100)
q = sympy.randprime(1, 100)

public, private = generate_keypair(p, q)

with open('keys.txt','w') as f:
    f.write(str(private[0]) + ' ' + str(public[0]) + ' ' + str(public[1]) + '\n') """

""" message = input("Type message: ")

encrypted_msg = encrypt(public, message)

print(f"{encrypted_msg}")

print (f"Decrypted Message is : {decrypt(private,encrypted_msg)}") """
```

4.4. Model building:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
import pickle

# loading the data from csv file to a pandas Dataframe
raw_mail_data = pd.read_csv('mail_data.csv')

print(raw_mail_data)

# replace the null values with a null string
mail_data = raw_mail_data.where((pd.notnull(raw_mail_data)), '')

# printing the first 5 rows of the dataframe
mail_data.head()
```

```
# checking the number of rows and columns in the dataframe
print(mail_data.shape)

# label spam mail as 0;  ham mail as 1;

mail_data.loc[mail_data['Category'] == 'spam', 'Category', ] = 0
mail_data.loc[mail_data['Category'] == 'ham', 'Category', ] = 1

# separating the data as texts and label

X = mail_data['Message']
Y = mail_data['Category']

print(X)
print(Y)

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=3)

print(X.shape)
print(X_train.shape)
print(X_test.shape)

# transform the text data to feature vectors that can be used as input to the Logistic regression

feature_extraction = TfidfVectorizer(min_df=1, stop_words='english', lowercase='True')

X_train_features = feature_extraction.fit_transform(X_train)
X_test_features = feature_extraction.transform(X_test)

# convert Y_train and Y_test values as integers

Y_train = Y_train.astype('int')
Y_test = Y_test.astype('int')

model = LogisticRegression()

# training the Logistic Regression model with the training data
model.fit(X_train_features, Y_train)

# prediction on training data

prediction_on_training_data = model.predict(X_train_features)
accuracy_on_training_data = accuracy_score(Y_train, prediction_on_training_data)

print('Accuracy on training data : ', accuracy_on_training_data)

# saving model using pickle
#pickle.dump(model, open('model.pkl', 'wb'))

# Save the vectorizer
```

```
vec_file = 'vectorizer.pickle'
#pickle.dump(feature_extraction, open(vec_file, 'wb'))
```

4.5. Predictor file:

```
 import pickle

  #load model
  model = pickle.load(open('model.pkl', 'rb'))

input_mail = ["I've been searching for the right words to thank you for this breather. I promise i
wont take your help for granted and will fulfil my promise. You have been wonderful and a blessing
at all times"]


# convert text to feature vectors and load vectorizer
   feature_extraction = pickle.load(open('vectorizer.pickle', 'rb'))
   input_data_features = feature_extraction.transform(input_mail_1)

# making prediction
   prediction = model.predict(input_data_features)
   print(prediction)

   if prediction[0] == 1:
       print('Ham mail')
   else:
       print('Spam mail')
```

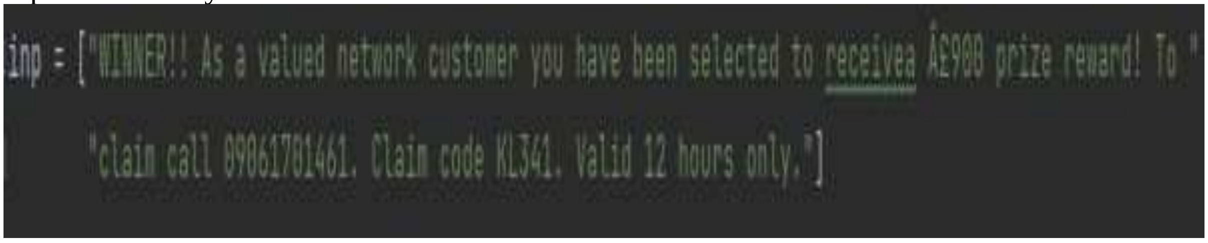
4.6 GUI Input file for SMTP:

```
#Import the required Libraries
from tkinter import *
from tkinter import ttk

#Create an instance of Tkinter frame
win= Tk()

#Set the geometry of Tkinter frame
win.geometry("750x250")

def display_text():
   global entry,win
   string= entry.get("1.0","end-1c")
   with open('mail_to_be_sent.txt','w') as f:
    f.write(string)
   win.destroy()

inp_lab = Label(win, text="Input mail:")
inp_lab.place(x=70,y=90)
```

```
label=Label(win, text="Developed by Inti Dhiraj", font=("Courier 12 bold"))
label.pack()

#Create an Entry widget to accept User Input
entry= Text(win,height=10, width= 40)
entry.focus_set()
entry.pack()

#Create a Button to validate Entry Widget
ttk.Button(win, text= "Send",width= 20, command= display_text).pack(pady=20)

win.mainloop()
```

5. SNAPSHOTS

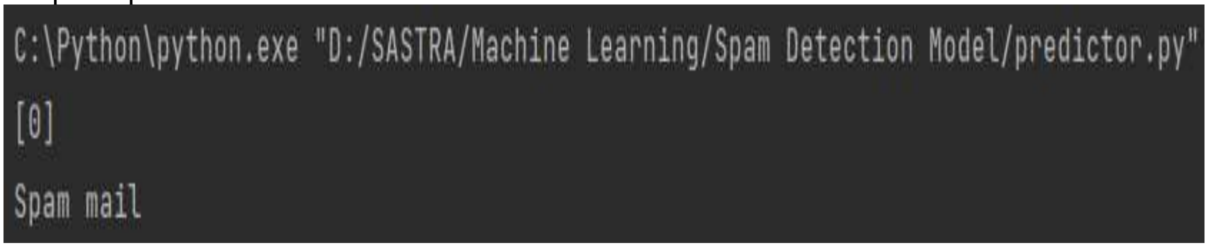
- Input: Mail body



```
inp = ["WINNER!! As a valued network customer you have been selected to receivea Â£900 prize reward! To "
       "claim call 09061701461. Claim code KL341. Valid 12 hours only."]
```

Fig 5.1 Input mail

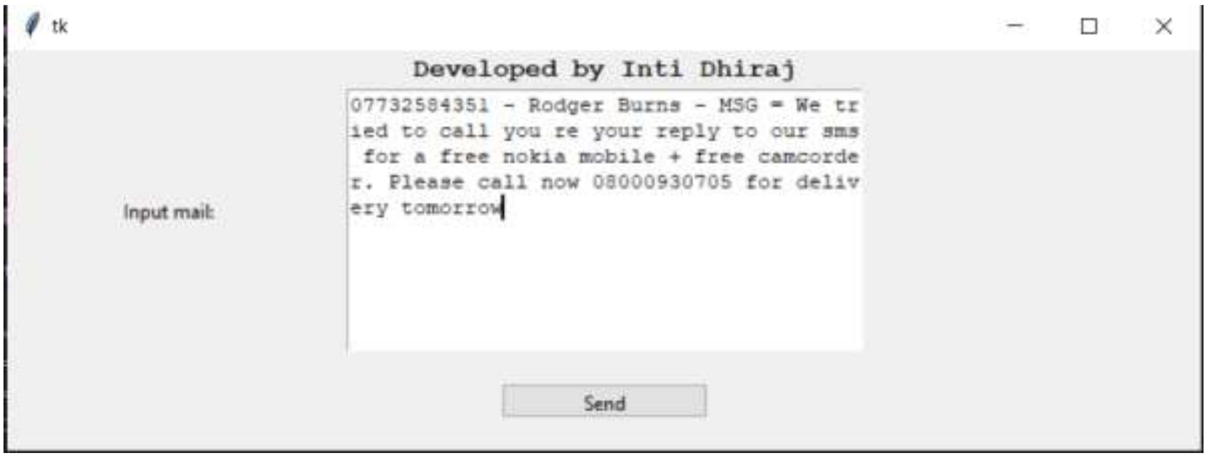
Output: Spam or Ham



```
C:\Python\python.exe "D:/SASTRA/Machine Learning/Spam Detection Model/predictor.py"
[0]
Spam mail
```

Fig 5.2 Output (Spam or Ham)

- Sending mail using SMTP module (Case - Spam): -



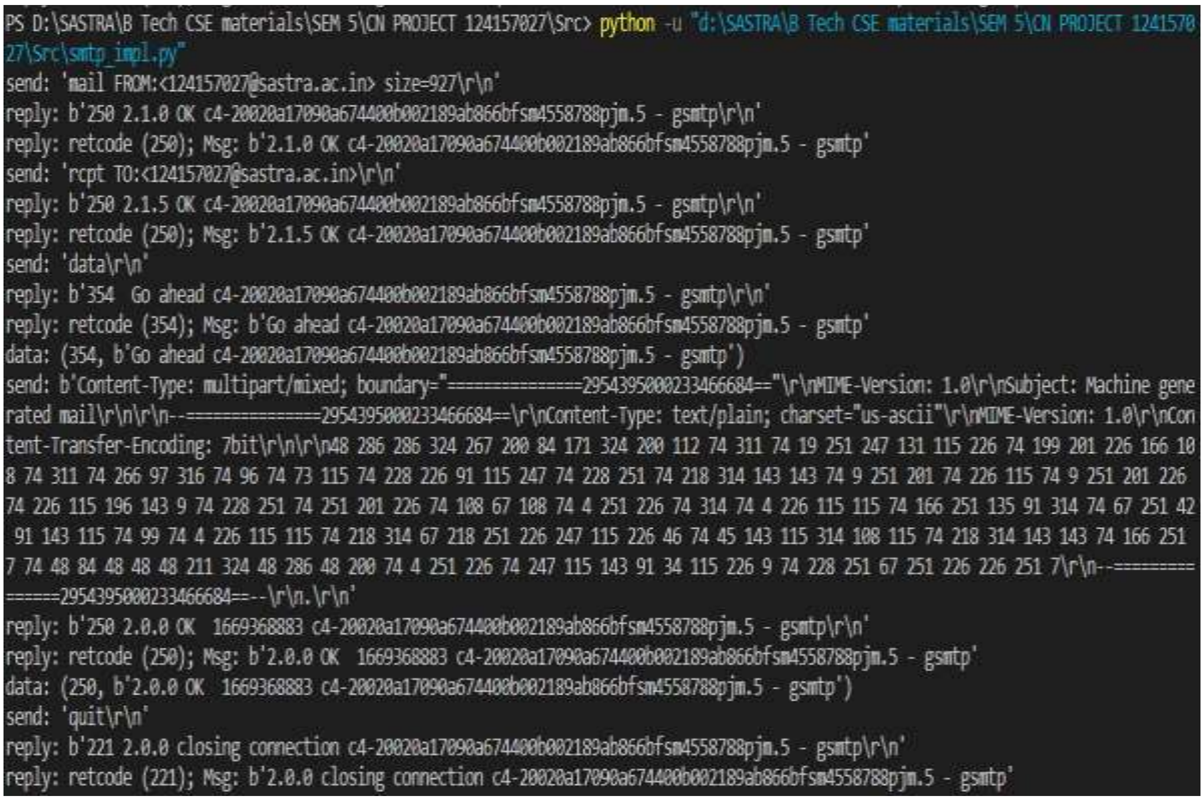
tk

Developed by Inti Dhiraj

Input mail:

```
07732584351 - Rodger Burns - MSG = We tr
ied to call you re your reply to our sms
 for a free nokia mobile + free camcorde
r. Please call now 08000930705 for deliv
ery tomorrow
```

Fig 5.3 GUI Sender’s side (Spam mail)

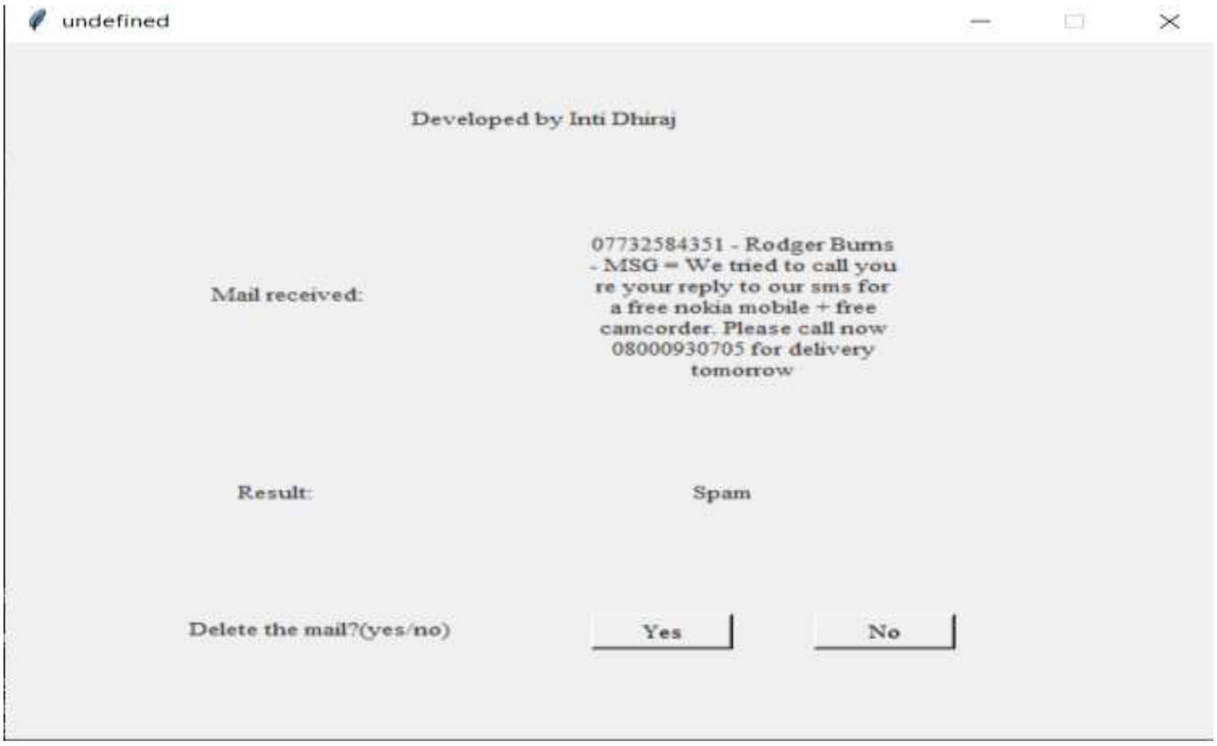


```
PS D:\SASTRA\B Tech CSE materials\SEM 5\CN PROJECT 124157027\Src> python -u "d:\SASTRA\B Tech CSE materials\SEM 5\CN PROJECT 1241570
27\Src\smtp_impl.py"
send: 'mail FROM:<124157027@sastra.ac.in> size=927\r\n'
reply: b'250 2.1.0 OK c4-20020a17090a674400b002189ab866bfsm4558788pjm.5 - gsmtp\r\n'
reply: retcode (250); Msg: b'2.1.0 OK c4-20020a17090a674400b002189ab866bfsm4558788pjm.5 - gsmtp'
send: 'rcpt TO:<124157027@sastra.ac.in>\r\n'
reply: b'250 2.1.5 OK c4-20020a17090a674400b002189ab866bfsm4558788pjm.5 - gsmtp\r\n'
reply: retcode (250); Msg: b'2.1.5 OK c4-20020a17090a674400b002189ab866bfsm4558788pjm.5 - gsmtp'
send: 'data\r\n'
reply: b'354  Go ahead c4-20020a17090a674400b002189ab866bfsm4558788pjm.5 - gsmtp\r\n'
reply: retcode (354); Msg: b'Go ahead c4-20020a17090a674400b002189ab866bfsm4558788pjm.5 - gsmtp'
data: (354, b'Go ahead c4-20020a17090a674400b002189ab866bfsm4558788pjm.5 - gsmtp')
send: b'Content-Type: multipart/mixed; boundary="===============2954395000233466684=="\r\nMIME-Version: 1.0\r\nSubject: Machine gene
rated mail\r\n\r\n--===============2954395000233466684==\r\nContent-Type: text/plain; charset="us-ascii"\r\nMIME-Version: 1.0\r\nCon
tent-Transfer-Encoding: 7bit\r\n\r\n48 286 286 324 267 200 84 171 324 200 112 74 311 74 19 251 247 131 115 226 74 199 201 226 166 10
8 74 311 74 266 97 316 74 96 74 73 115 74 228 226 91 115 247 74 228 251 74 218 314 143 143 74 9 251 201 74 226 115 74 9 251 201 226
74 226 115 196 143 9 74 228 251 74 251 201 226 74 108 67 108 74 4 251 226 74 314 74 4 226 115 115 74 166 251 135 91 314 74 67 251 42
 91 143 115 74 99 74 4 226 115 115 74 218 314 67 218 251 226 247 115 226 46 74 45 143 115 314 108 115 74 218 314 143 143 74 166 251
7 74 48 84 48 48 48 211 324 48 286 48 200 74 4 251 226 74 247 115 143 91 34 115 226 9 74 228 251 67 251 226 226 251 7\r\n--=========
======2954395000233466684==--\r\n.\r\n'
reply: b'250 2.0.0 OK  1669368883 c4-20020a17090a674400b002189ab866bfsm4558788pjm.5 - gsmtp\r\n'
reply: retcode (250); Msg: b'2.0.0 OK  1669368883 c4-20020a17090a674400b002189ab866bfsm4558788pjm.5 - gsmtp'
data: (250, b'2.0.0 OK  1669368883 c4-20020a17090a674400b002189ab866bfsm4558788pjm.5 - gsmtp')
send: 'quit\r\n'
reply: b'221 2.0.0 closing connection c4-20020a17090a674400b002189ab866bfsm4558788pjm.5 - gsmtp\r\n'
reply: retcode (221); Msg: b'2.0.0 closing connection c4-20020a17090a674400b002189ab866bfsm4558788pjm.5 - gsmtp'
```

Fig 5.4 Communication with SMTP server

Actual mail body = "07732584351 - Rodger Burns - MSG = We tried to call you re your reply to our sms for a free nokia mobile + free camcorder. Please call now 08000930705 for delivery tomorrow"

- Receiving mail using IMAP module: -



undefined

Developed by Inti Dhiraj

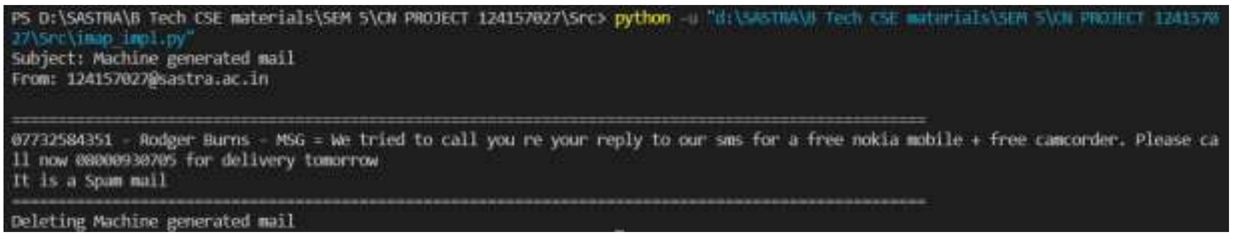
Mail received: 07732584351 - Rodger Burns - MSG = We tried to call you re your reply to our sms for a free nokia mobile + free camcorder. Please call now 08000930705 for delivery tomorrow

Result: Spam

Delete the mail?(yes/no)

Fig 5.5 GUI to show whether spam or ham

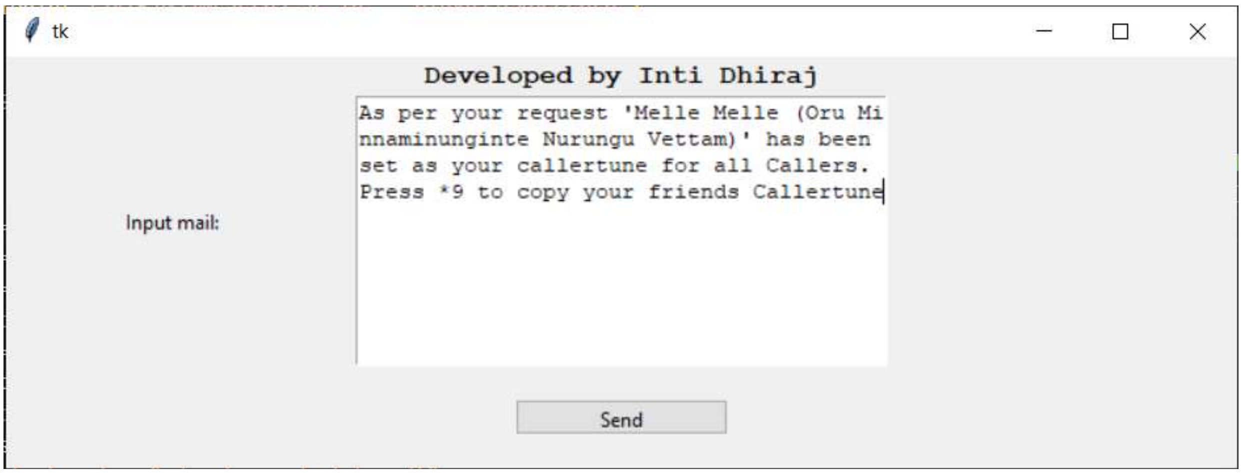
Clicked yes in the above dialogue box



```
PS D:\SASTRA\B Tech CSE materials\SEM 5\CN PROJECT 124157027\Src> python -u "d:\SASTRA\B Tech CSE materials\SEM 5\CN PROJECT 1241570
27\Src\imap_impl.py"
Subject: Machine generated mail
From: 124157027@sastra.ac.in

==================================================================================================
07732584351 - Rodger Burns - MSG = We tried to call you re your reply to our sms for a free nokia mobile + free camcorder. Please ca
ll now 08000930705 for delivery tomorrow
It is a Spam mail
==================================================================================================
Deleting Machine generated mail
```

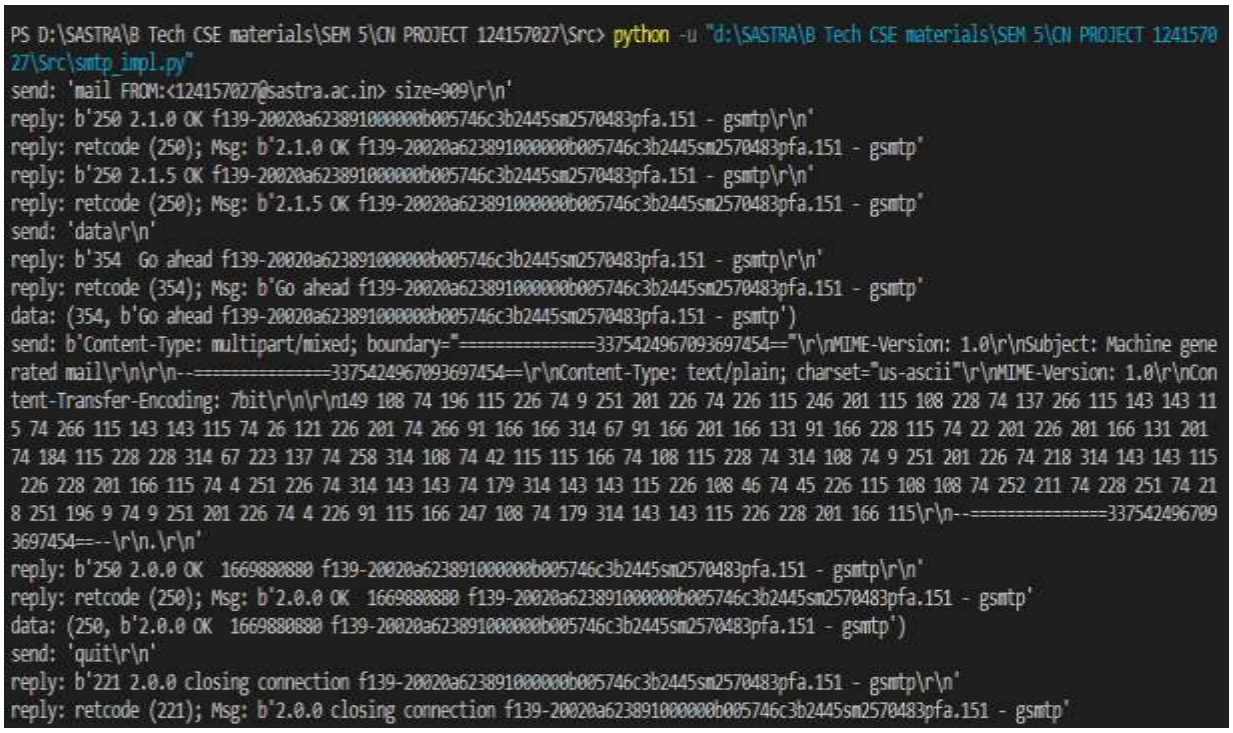
Fig 5.6 IMAP module output

- Sending mail using SMTP module (Case – Ham): -



```
As per your request 'Melle Melle (Oru Mi
nnaminunginte Nurungu Vettam)' has been
set as your callertune for all Callers.
Press *9 to copy your friends Callertune
```

Fig 5.7 GUI Sender's side (Ham mail)

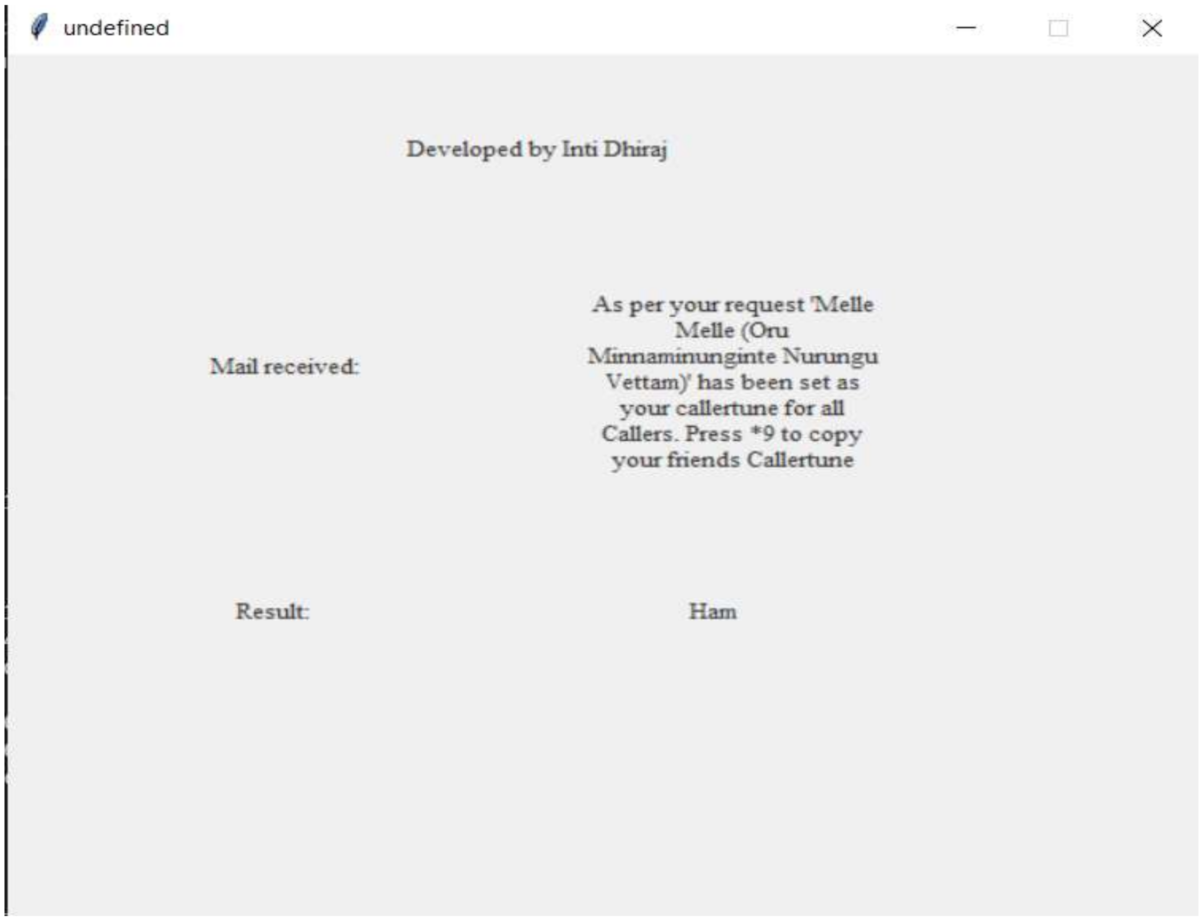


```
PS D:\SASTRA\B Tech CSE materials\SEM 5\CN PROJECT 124157027\Src> python -u "d:\SASTRA\B Tech CSE materials\SEM 5\CN PROJECT 1241570
27\Src\smtp_impl.py"
send: 'mail FROM:<124157027@sastra.ac.in> size=909\r\n'
reply: b'250 2.1.0 OK f139-20020a623891000000b005746c3b2445sm2570483pfa.151 - gsmtp\r\n'
reply: retcode (250); Msg: b'2.1.0 OK f139-20020a623891000000b005746c3b2445sm2570483pfa.151 - gsmtp'
reply: b'250 2.1.5 OK f139-20020a623891000000b005746c3b2445sm2570483pfa.151 - gsmtp\r\n'
reply: retcode (250); Msg: b'2.1.5 OK f139-20020a623891000000b005746c3b2445sm2570483pfa.151 - gsmtp'
send: 'data\r\n'
reply: b'354  Go ahead f139-20020a623891000000b005746c3b2445sm2570483pfa.151 - gsmtp\r\n'
reply: retcode (354); Msg: b'Go ahead f139-20020a623891000000b005746c3b2445sm2570483pfa.151 - gsmtp'
data: (354, b'Go ahead f139-20020a623891000000b005746c3b2445sm2570483pfa.151 - gsmtp')
send: b'Content-Type: multipart/mixed; boundary="===============3375424967093697454=="\r\nMIME-Version: 1.0\r\nSubject: Machine gene
rated mail\r\n\r\n--===============3375424967093697454==\r\nContent-Type: text/plain; charset="us-ascii"\r\nMIME-Version: 1.0\r\nCon
tent-Transfer-Encoding: 7bit\r\n\r\n149 108 74 196 115 226 74 9 251 201 226 74 226 115 246 201 115 108 228 74 137 266 115 143 143 11
5 74 266 115 143 143 115 74 26 121 226 201 74 266 91 166 166 314 67 91 166 201 166 131 91 166 228 115 74 22 201 226 201 166 131 201
74 184 115 228 228 314 67 223 137 74 258 314 108 74 42 115 115 166 74 108 115 228 74 314 108 74 9 251 201 226 74 218 314 143 143 115
 226 228 201 166 115 74 4 251 226 74 314 143 143 74 179 314 143 143 115 226 108 46 74 45 226 115 108 108 74 252 211 74 228 251 74 21
8 251 196 9 74 9 251 201 226 74 4 226 91 115 166 247 108 74 179 314 143 143 115 226 228 201 166 115\r\n--===============337542496709
3697454==--\r\n.\r\n'
reply: b'250 2.0.0 OK  1669880880 f139-20020a623891000000b005746c3b2445sm2570483pfa.151 - gsmtp\r\n'
reply: retcode (250); Msg: b'2.0.0 OK  1669880880 f139-20020a623891000000b005746c3b2445sm2570483pfa.151 - gsmtp'
data: (250, b'2.0.0 OK  1669880880 f139-20020a623891000000b005746c3b2445sm2570483pfa.151 - gsmtp')
send: 'quit\r\n'
reply: b'221 2.0.0 closing connection f139-20020a623891000000b005746c3b2445sm2570483pfa.151 - gsmtp\r\n'
reply: retcode (221); Msg: b'2.0.0 closing connection f139-20020a623891000000b005746c3b2445sm2570483pfa.151 - gsmtp'
```

Fig 5.8 Communication with SMTP server

Actual mail body = "As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your callertune for all Callers. Press *9 to copy your friends Callertune "

- Receiving mail using IMAP module: -



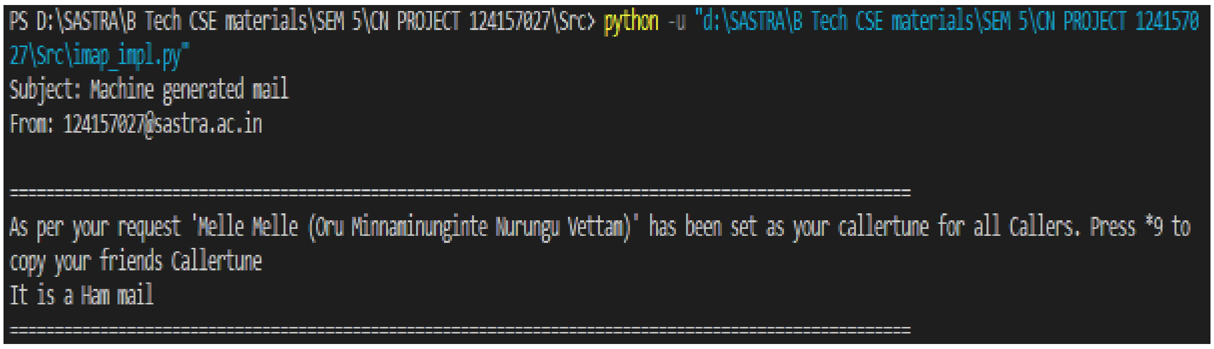
undefined

Developed by Inti Dhiraj

Mail received: As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your callertune for all Callers. Press *9 to copy your friends Callertune

Result: Ham

Fig 5.9 GUI to show whether spam or ham



```
PS D:\SASTRA\B Tech CSE materials\SEM 5\CN PROJECT 124157027\Src> python -u "d:\SASTRA\B Tech CSE materials\SEM 5\CN PROJECT 1241570
27\Src\imap_impl.py"
Subject: Machine generated mail
From: 124157027@sastra.ac.in

==========================================================================================
As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your callertune for all Callers. Press *9 to
copy your friends Callertune
It is a Ham mail
==========================================================================================
```

Fig 5.10 IMAP module output

6. CONCLUSION & FUTURE WORK

6.1. Conclusion:

Mailing is very common nowadays, working on this project has led to development of an application which can remotely log in to the email account and perform various functions such as sending a mail or fetching a mail and also classifying it as a spam or not. Also, this application improves confidentiality and authenticity while communicating over third-party email servers, moreover the user's experience is enriched, and hence life is made easy. This project is also helpful in understanding the simple mechanism of mail transfer protocols and RSA cryptosystem.

6.2. Future Work:

1) Other functionalities like regular cleanup can be implemented.
2) Post quantum cryptography may be implemented to increase the security of the cryptosystem employed.
3) Develop own server, with a registered domain name, i.e., setting up SMTP and IMAP server for email users.

7. REFERENCES

- A. Karim, "Comprehensive Survey for Intelligent Spam Email Detection", December 4, 2019).
- Maryam Hina, Mohsan Ali, Abdul Rehman Javed, Gautam Srivastava, Thippa Reddy Gadekallu, Zunera Jalil, "Email Classification and Forensics Analysis using Machine Learning", 2021.
- https://realpython.com/python-send-email/
- https://www.youtube.com/watch?v=rxkGItX5gGE
- https://www.techtarget.com/whatis/definition/MIME-Multi-Purpose-Internet-Mail-Extensions#:~:text=MIME%20(Multipurpose%20Internet%20Mail%20Extensions)%20is%20an%20extension%20of%20the,and%20application%20programs%2C%20over%20email.
- https://www.techtarget.com/whatis/definition/IMAP-Internet-Message-Access-Protocol#:~:text=The%20headers%20of%20all%20emails,Office%20Protocol%203%20(POP3).