**Req_159**: Email shall be sent to any valid email address from To, Cc, and/or Bcc address lines
Assumptions:

1) User email is user@test.com, test control emails are test1@test.com, test2@test.com, and so on.

Test Procedure:

Step 1: Log into/ identify the server for testing.

Step 2: Run the testing code send email with the correct server.

Step 3: Ensure the code runs properly and reports that there were errors in sending to the invalid_address emails only.

Test Automation:

```
import smtplib, ssl
import imaplib
import email
import os

def send_email(imap_ssl_host):

        # Accessing the server
        server = imaplib.IMAP4_SSL(imap_ssl_host)

        # Identifying the emails to send to including invalid emails that it should not send to
        user_email = user@test.com
        send_to =['test1@test.com,test1a@test.com,invalid_address1@com']
        send_cc =['test2@test.com,test2a@test.com,invalid_address2.com']
        send_bcc =['test3@test.com,test3a@test.com,invalid_address3']

        # Creating the message and data for the message
        message= MIMEMultipart('alternative')
        message['Body']="""\This is a test."""
        message['Subject']="Test"
        message['To']=send_to
        message['Cc']=send_cc
        message['Bcc']=send_bcc

        # Splitting the addresses and combining into one list for sending
        send_total= send_to.split(',') + send_cc.split(',') + send_bcc.split(',')
```

```
        # Sending the message
        try:
                server.sendmail(user_email, send_total, message.as.string())

        #Checking that the message is sent but only to the valid email addresses
        except ssl.SMTPConnectionError:
                print(_logger.exception('SMTP Connection failed to %s on %s' % (host, port)))
        except Exception as e:
                 print(e)
```

**Req_42:** All email addresses identified as spam by the user shall be automatically sent to the spam folder

Assumptions:

1) User email is user@test.com, test control email is test@test.com and spam test email is spam@test.com. The user has identified spam@test.com as spam.
2) The backend application has access to the folders spam and inbox.

Test Procedure:

Step 1: Log into the user@test.com email.

Step 2: Identify spam@test.com as a spam email address.

Step 3: Run the spam_test function with local server.

Step 4: Ensure that no error messages print and that “Test email was received in inbox” and “Spam email was received in spam” are returned.

Test Automation:

```
import smtplib, ssl
import imaplib
import email
import os


def spam_test(imap_ssl_host):
        server = imaplib.IMAP4_SSL(imap_ssl_host)

        # Sending test emails that should reach inbox and spam
        sending_email = 'test@test.com'
        spam_email = 'spam@test.com'
        receiving_email = 'user@test.com'
```

```
message= MIMEMultipart(‘alternative’)
message[‘Body’] = """\This is a test."""
message[‘Subject’]=”Test”
message_spam= MIMEMultipart(‘alternative’)
message_spam[‘Body’] = """\This is a test for spam."""
message_spam[‘Subject’]=”Test Spam”

try:
        server.sendmail(sending_email, receiving_email, message.as_string())
        server.sendmail(spam_email, receiving_email, message_spam.as_string())
except Exception as e:
        print(e)

# Access receiving email
server.login(receiving_email,  password)

# Accessing the Inbox
server.select(‘Inbox’)

# Checking if the test email was properly received in the inbox, printing error if not
result, data = mail.search(None, '(FROM "test@test.com")')
if data:
        print(“Test email was received in inbox”)
else:
        print(“Error: Test email was NOT received in inbox”)
# Checking if the spam test email incorrectly ended up in inbox
result, data = mail.search(None, '(FROM "spam@test.com")')
if data:
        print(“Error: Spam email received in inbox”)
# Switching to the spam folder
server.select(‘Spam’)
# Checking if the spam test email was properly received in spam, printing error if not
result, data = mail.search(None, '(FROM "spam@test.com")')
if data:
        print(“Spam email was received in spam”)
else:
        print(“Error: Spam email was NOT received in spam”)
# Checking if the test email incorrectly ended up in spam
result, data = mail.search(None, '(FROM "test@test.com")')
if data:
        print(“Error: Test email received in spam”)
```