



E-MAIL APP

An Email Application using Java Swing and JavaMail

Abstract

Electronic mail (email or e-mail) is a method of exchanging messages ("mail") between people using electronic devices. In this document we demonstrate how an E-mail application can be developed using Java Swing and JavaMail Api . Features include composing mails, inbox, sent mails ,read-ceipts and viewing mails. For simplicity purposes, instead of building our own SMTP server , we use Gmail's SMTP server .

Amartya Choudhury
JU-BCSE IV
Roll-001610501026
Internet Technology Laboratory
amartyachowdhuy98@gmail.com

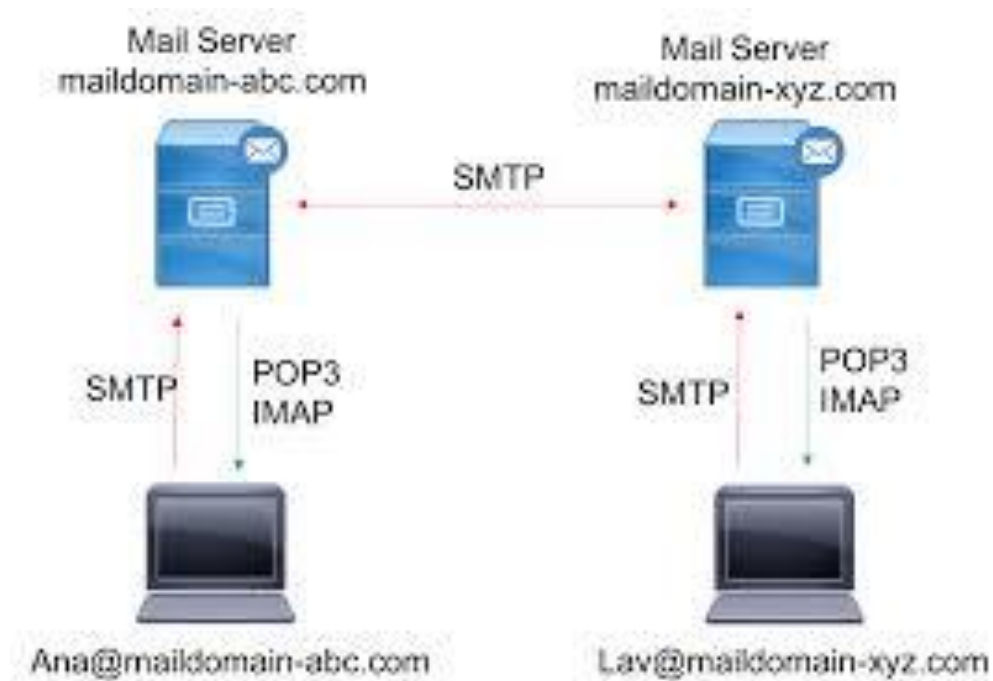
Problem Statement:

Design a Mail server and client program that implements SMTP and POP3 protocols for sending and receiving emails. An acknowledgement is sent to the sender whenever the recipient views the email. Recipient should find email notifications whenever (s)he logs in. Two possible use cases are shown above. For the details of POP3 and SMTP protocols you can refer to RFC 1939

(<http://tools.ietf.org/html/rfc1939>) and RFC 821 (<http://tools.ietf.org/html/rfc821>), respectively

Observe corresponding packet flow using Wireshark and report on traffic analysis.

E-Mail Architecture:



Design:

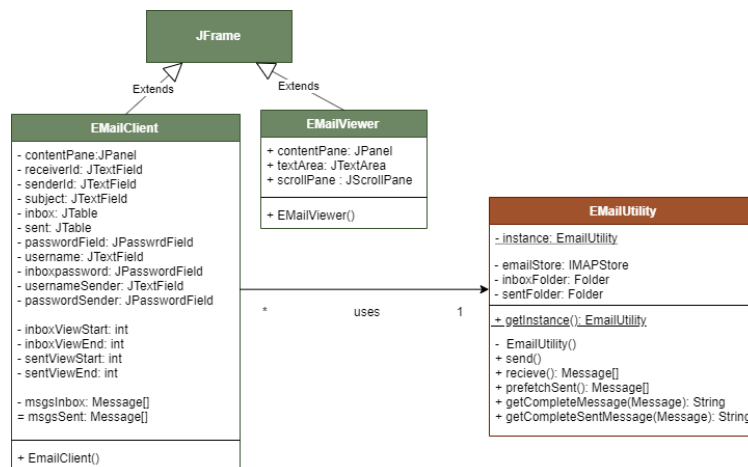
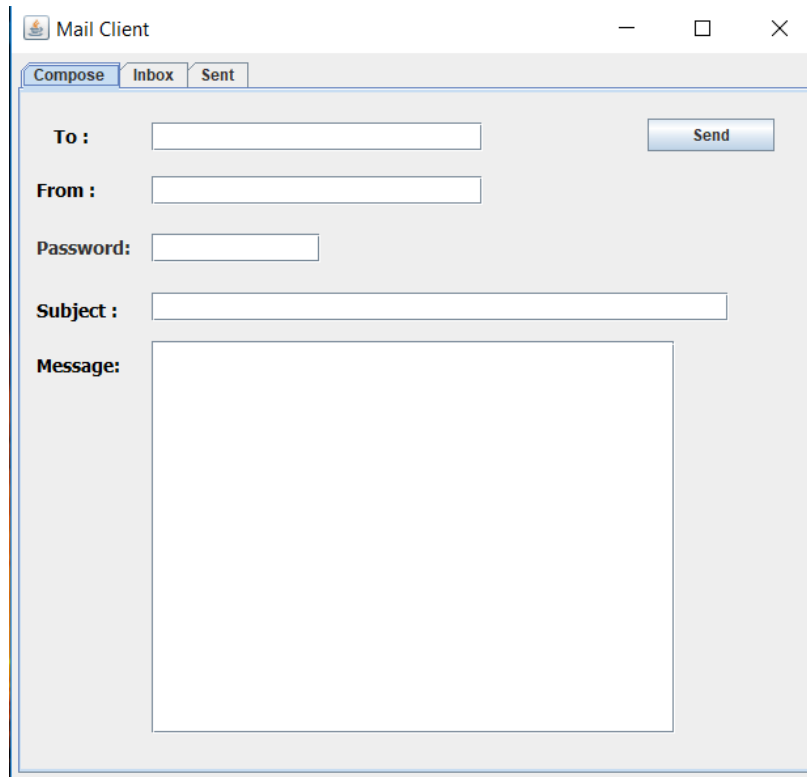


Fig 1: Class Diagram

Tool:



The image shows a screenshot of a 'Mail Client' window. The window has a title bar with the text 'Mail Client' and standard minimize, maximize, and close buttons. Below the title bar, there are three tabs: 'Compose', 'Inbox', and 'Sent'. The 'Compose' tab is currently selected. The main area of the window is a form for composing an email. It includes labels and input fields for 'To:', 'From:', 'Password:', and 'Subject:'. There is a 'Send' button located to the right of the 'To:' field. Below these fields is a large text area labeled 'Message:'. The form is set against a light gray background.

Mail Client

Compose Inbox Sent

To :

From :

Password:

Subject :

Message:

Send

Fig 2: Compose Panel

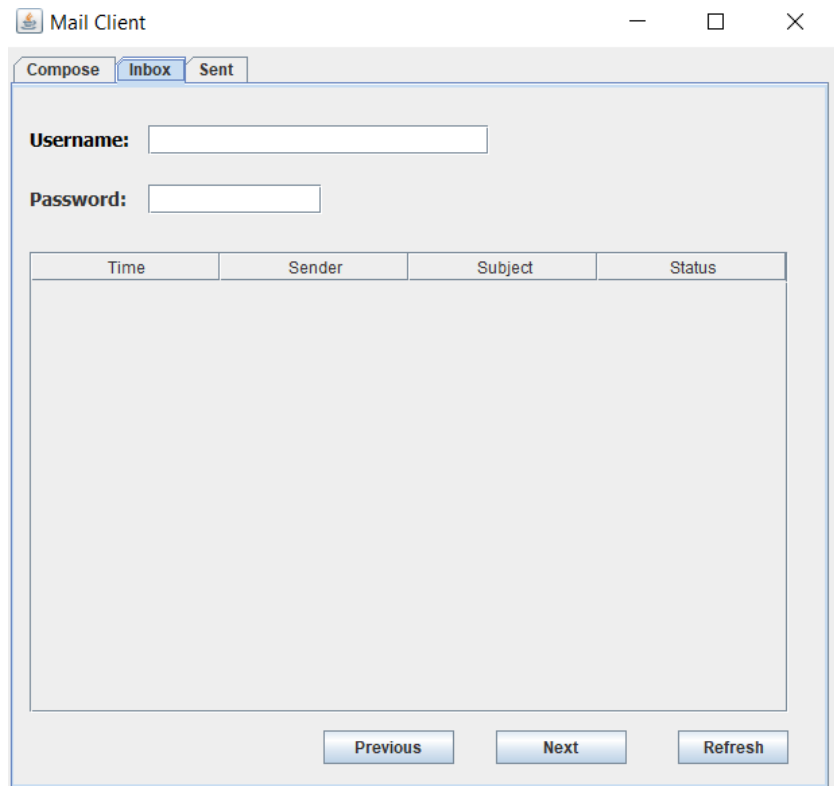


Fig 3: Inbox Panel

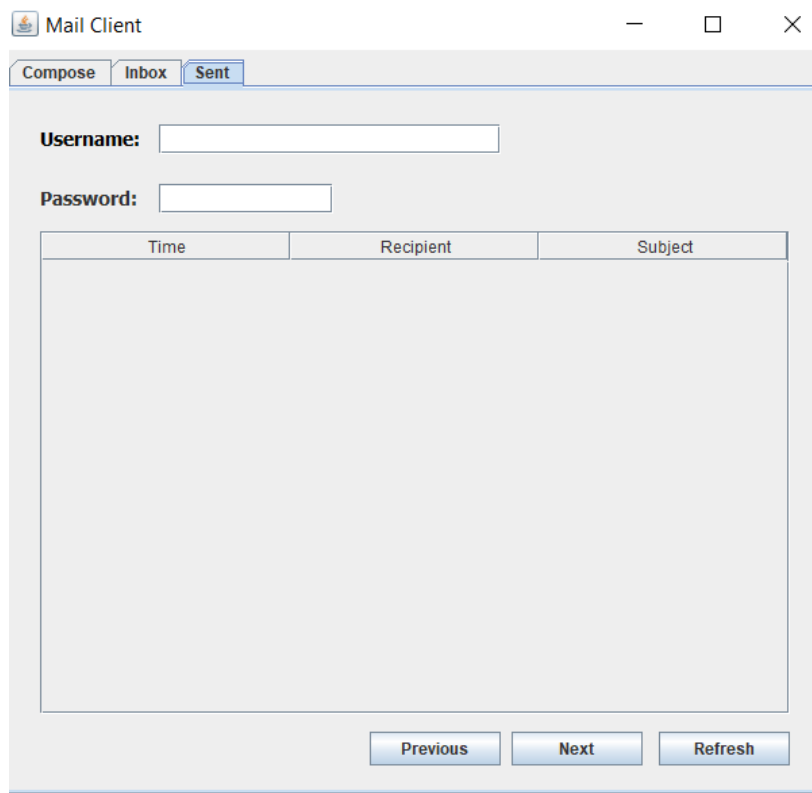


Fig 4 : Sent Mail Panel

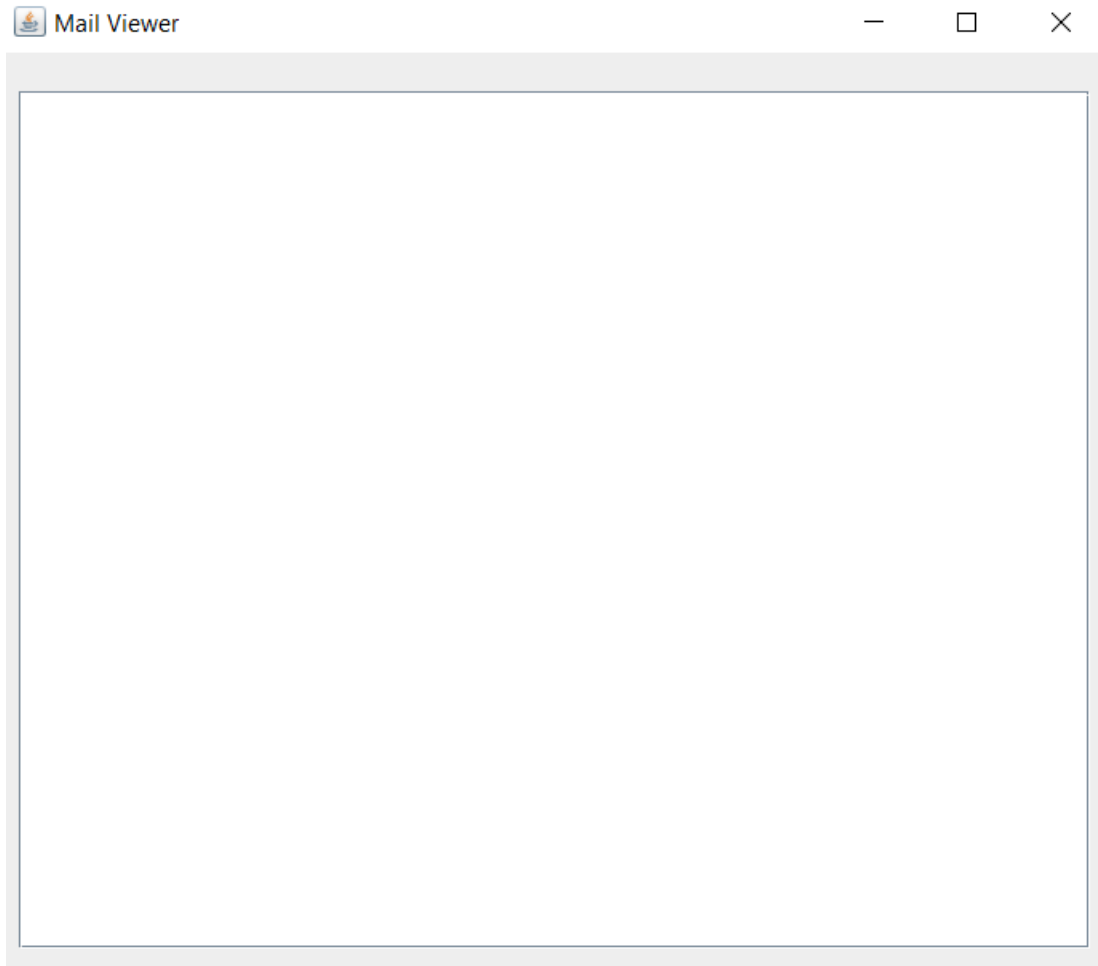


Fig 5 : Mail Viewer

Workflow:

Message Sending:

1. When the user clicks “Send” from the compose panel, the fields are extracted from the form and sent as parameters to the “send” function EmailUtility , a singleton.
2. After sender username and password are authenticated, a new session is formed with the Gmail SMTP server

3. An instance of "MimeMessage" is created, and recipient is added. The mail subject and text added . A disposition notification is added to the message header requesting read receipts.
4. The message is sent .

EmailClient :

```
JButton btnSend = new JButton("Send");
btnSend.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        String to = recieverId.getText();
        String from = senderId.getText();
        String password = passwordField.getText();
        String sub = subject.getText();
        String msg = messageArea.getText();
        EmailUtility.getInstance().send(from, password, to, sub, msg);
    }
});
```

EmailUtility :

```
public void send(String from,String password,String to,String sub,String msg) {
    Properties props = new Properties();
    props.put("mail.smtp.host","smtp.gmail.com");
    props.put("mail.smtp.auth","true");
    props.put("mail.smtp.socketFactory.port", "465");
    props.put("mail.smtp.socketFactory.class","javax.net.ssl.SSLSocketFactory");
    props.put("mail.smtp.port","465");

    Session session = Session.getInstance(props,
        new javax.mail.Authenticator() {
            protected PasswordAuthentication getPasswordAuthentication()
            {
                return new PasswordAuthentication(from,password);
            }
        });

    try {
        MimeMessage message = new MimeMessage(session);
        message.addRecipient(Message.RecipientType.TO,new
InternetAddress(to));
        message.setSubject(sub);
        message.setText(msg);
        message.setHeader("Disposition-Notification-To",from);
    }
}
```

```

        //send message
        Transport.send(message);
        System.out.println("message sent successfully");
    }
    catch (MessagingException e) {
        throw new RuntimeException(e);
    }
    catch (Exception e) {

        throw new RuntimeException(e);
    }
    return;
}

```

Message Inbox:

- When “Refresh” button of the inbox panel is clicked ,mose recent 100 messages are prefetched from the INBOX Folder of the Server . The next 100 messages can be prefeteched by using the “Next” button .
- For prefetching , a FetchProfile object is used which specifies which parts of the mail need only be fetched (We fetch the ENVELOPE,FLAGS and CONTENT_INFO).
- The prefetched data for a mimemessage has the date , subject , sender and Flags (SEEN,RECENT etc).
- The messages are displayed in the inbox panel in a Jtable .
- On clicking a particular message of interest , an instance of “EmailViewer” is created ,and the whole MimeMessage is fetched using the getCompleteMessage() function of the EmailUtility class . The message is displayed in the JTextArea of the EmailViewer.
- The SEEN Flag associated with the message gets set when the whole message is pulled from the server (message.getContent())
- When the whole message pulled, a check is done whether the message has a “Disposition-Notification” required header .If present , then a new message with “Multipart/Report” content type is sent back to the sender.

EmailClient :

```
Jbutton btnRefreshInbox = new Jbutton("Refresh");
    btnRefreshInbox.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            String user = username.getText();
            String password = inboxpassword.getText();
            msgsInbox = EmailUtility.getInstance().recieve(user,
password,inboxViewStart,inboxViewEnd);
            if(msgsInbox != null) {
                DefaultTableModel inboxmodel =
(DefaultTableModel)inbox.getModel();
                int rowCount = inboxmodel.getRowCount();
                for (int i = rowCount - 1; i >= 0; i--) {
                    inboxmodel.removeRow(i);
                }
                for(int i= msgsInbox.length -1;i>=0;i--) {
                    Message msg = msgsInbox[i];
                    try {

                        if(msg.getFlags().contains(Flags.Flag.RECENT)) inboxmodel.addRow(new Object[]
{msg.getSentDate(),msg.getFrom()[0],msg.getSubject(),"RECENT"});
                            else
if(msg.getFlags().contains(Flags.Flag.SEEN)) inboxmodel.addRow(new Object[]
{msg.getSentDate(),msg.getFrom()[0],msg.getSubject(),"SEEN"});
                                else inboxmodel.addRow(new
Object[] {msg.getSentDate(),msg.getFrom()[0],msg.getSubject(),"UNREAD"});
                            } catch (MessagingException e1) {
                                // TODO Auto-generated catch
block
                                e1.printStackTrace();
                            }
                        }
                    }
                }
            });
```

EmailUtility :

```
public Message[] recieve(String user,String password,int start,int end) {
    Message[] messages = null;
    try {
        Properties properties = new Properties();
```

```

        properties.put("mail.imap.host",
"imap.gmail.com");

        properties.put("mail.imap.ssl.enable", "true");
        properties.put("mail.imap.port", "993");
        Session emailSession =
Session.getInstance(properties,

        new javax.mail.Authenticator() {
            protected PasswordAuthentication
getPasswordAuthentication() {

                return new
PasswordAuthentication(user,password);
            }
        });

        emailStore = (IMAPStore)
emailSession.getStore("imap");
        emailStore.connect(user,password);
        inboxFolder = emailStore.getFolder("INBOX");
        inboxFolder.open(Folder.READ_ONLY);
        System.out.println(inboxFolder.getMessageCount());
        messages =
inboxFolder.getMessages(inboxFolder.getMessageCount() -end +
1,inboxFolder.getMessageCount() - start + 1);
        FetchProfile profile = new FetchProfile();
        profile.add(FetchProfileItem.ENVELOPE);
        profile.add(FetchProfileItem.FLAGS);
        profile.add(FetchProfileItem.CONTENT_INFO);
        profile.add("X-mailer");
        inboxFolder.fetch(messages, profile);
        System.out.println("fetching done");
        inboxFolder.close(false);
        //emailStore.close();

    }
    catch (NoSuchProviderException e)
{e.printStackTrace();}

    catch (MessagingException e) {e.printStackTrace();}
    catch (Exception e) {e.printStackTrace();}

    return messages;
}

public String getCompleteMessage(Message msg,String from,String
password) {
    String email = "";
    try {

```

```

        email += "\n";
    }
    email += "From : " + msg.getFrom()[0] + "\n";
    email += "To : " +
msg.getRecipients(Message.RecipientType.TO)[0] + "\n";
    email += "Date : " + msg.getSentDate().toString() +
"\n";
    email += "Subject : " + msg.getSubject() + "\n";
    email += "\n";
}

inboxFolder.open(Folder.READ_WRITE);
System.out.println(msg.getContentType());
if (msg.isMimeType("text/plain")) {
    email += msg.getContent().toString();
    if(msg.getHeader("Disposition-Notification-
To") != null) {

        System.out.println(msg.getHeader("Disposition-Notification-To")[0]);
        Properties props = new Properties();
        props.put("mail.smtp.host", "smtp.gmail.com");
        props.put("mail.smtp.auth", "true");
        props.put("mail.smtp.socketFactory.port",
"465");

        props.put("mail.smtp.socketFactory.class", "javax.net.ssl.SSLSocketFactory");
        props.put("mail.smtp.port", "465");

        Session session = Session.getInstance(props,
            new javax.mail.Authenticator() {
                protected PasswordAuthentication
getPasswordAuthentication() {
                    return new
PasswordAuthentication(from,password);
                }
            });

        DispositionNotification dn = new
DispositionNotification();

        MultipartReport mr = new MultipartReport("---
-----Received your mail : " +msg.getSubject() + "-----", dn);

```

```

        try {
            MimeMessage message = new
MimeMessage(session);

message.addRecipient(Message.RecipientType.TO, new
InternetAddress(msg.getHeader("Disposition-Notification-To")[0]));
            message.setContent(mr);
            message.setSubject("DISPOSITION-
NOTIFICATION");

            Transport.send(message);
            System.out.println("mdn sent
successfully");
        }
        catch (MessagingException e) {
            throw new RuntimeException(e);
        }
        catch (Exception e) {

            throw new RuntimeException(e);
        }
    }
}

else if (msg.isMimeType("multipart/*")) {
    String result = "";
    if (msg.isMimeType("multipart/report")) {
        MultipartReport report = (MultipartReport)
msg.getContent();

        int count = report.getCount();
        for (int i = 0; i < count; i++) {
            BodyPart bodyPart =
report.getBodyPart(i);

            if
(bodyPart.isMimeType("text/plain")) {

                result = result + "\n" +
bodyPart.getContent().toString();

                break;
            }
        }
        email += result;
    }
    else {
        MimeMultipart mimeMultipart =
(MimeMultipart)msg.getContent();
        int count = mimeMultipart.getCount();

```

```

        for (int i = 0; i < count; i++){
            BodyPart bodyPart =
mimeMultipart.getBodyPart(i);

            if
(bodyPart.isMimeType("text/plain")){

                result = result + "\n" +
bodyPart.getContent().toString();

                break;
            }
        }
        email += result;
        if (msg.getHeader("Disposition-Notification-To") !=
null) {

            System.out.println(msg.getHeader("Disposition-Notification-To")[0]);
            Properties props = new Properties();
            props.put("mail.smtp.host", "smtp.gmail.com");
            props.put("mail.smtp.auth", "true");
            props.put("mail.smtp.socketFactory.port",
"465");

            props.put("mail.smtp.socketFactory.class", "javax.net.ssl.SSLSocketFactory");
            props.put("mail.smtp.port", "465");

            Session session = Session.getInstance(props,
                new javax.mail.Authenticator() {
                    protected PasswordAuthentication
getPasswordAuthentication() {
                        return new
PasswordAuthentication(from,password);
                    }
                });

            DispositionNotification dn = new
DispositionNotification();

            MultipartReport mr = new MultipartReport("---
-----Received your mail : " +msg.getSubject() + "-----",dn);

            try {
                MimeMessage message = new
MimeMessage(session);

```

```

message.addRecipient(Message.RecipientType.TO, new
InternetAddress(msg.getHeader("Disposition-Notification-To")[0]));
        message.setContent(mr);
        message.setSubject("DISPOSITION-
NOTIFICATION");

        Transport.send(message);
        System.out.println("mdn sent
successfully");

    }
    catch (MessagingException e) {
        throw new RuntimeException(e);
    }
    catch (Exception e) {

        throw new RuntimeException(e);
    }
}
}
}
    inboxFolder.close(false);
}
catch (MessagingException e) {e.printStackTrace();}
catch (IOException e) { e.printStackTrace();}
return email;
}

```

Sent messages:

- Sent messages are also pulled from the SMTP server's "SENT" Folder, starting with the top 100 most recent. They are prefetched and info. Like "Recipient", Date, Subject is displayed. Message is pulled when clicked, and displayed in an "EmailViewer" .

EmailClient :

```

Jbutton btnRefreshSent = new Jbutton("Refresh");
    btnRefreshSent.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            String username = usernameSender.getText();
            String password = passwordSender.getText();

```

```

        msgsSent =
EmailUtility.getInstance().prefetchSent(username,
password,sentViewStart,sentViewEnd);
        if(msgsSent != null) {
            DefaultTableModel sentmodel =
(DefaultTableModel)sent.getModel();
            int rowCount = sentmodel.getRowCount();
            for (int i = rowCount - 1; i >= 0; i--) {
                sentmodel.removeRow(i);
            }
            for(int i= msgsSent.length -1;i>=0;i--) {
                Message msg = msgsSent[i];
                try {

                    if(msg.getFlags().contains(Flags.Flag.RECENT)) sentmodel.addRow(new Object[]
{msg.getSentDate(),msg.getRecipients(RecipientType.TO)[0],msg.getSubject()});
                    else
                    if(msg.getFlags().contains(Flags.Flag.SEEN)) sentmodel.addRow(new Object[]
{msg.getSentDate(),msg.getRecipients(RecipientType.TO)[0],msg.getSubject()});
                    else sentmodel.addRow(new
Object[] {msg.getSentDate(),msg.getFrom()[0],msg.getSubject()});
                } catch (MessagingException e1) {
                    // TODO Auto-generated catch
                    e1.printStackTrace();
                }
            }
        }
    }
});

```

EmailUtility :

```

public String getCompleteSentMessage(Message msg) {
    String email = "";
    try {
        email += "-----\n";
        email += "-----\n";
        email += "From :" + msg.getFrom()[0]+"\n";
        email += "To :"+
msg.getRecipients(Message.RecipientType.TO)[0] +"\n";
        email += "Date :"+ msg.getSentDate().toString() +
"\n";
        email += "Subject :"+ msg.getSubject() + "\n";
    }
}

```

```

        email += "\n";
    }

    sentFolder.open(Folder.READ_ONLY);
    System.out.println(msg.getContentType());
    if (msg.isMimeType("text/plain")){
        email += msg.getContent().toString();
    }

    else if (msg.isMimeType("multipart/*")) {
        String result = "";
        MimeMultipart mimeMultipart =
(MimeMultipart)msg.getContent();
        int count = mimeMultipart.getCount();
        for (int i = 0; i < count; i++){
            BodyPart bodyPart =
mimeMultipart.getBodyPart(i);
            if (bodyPart.isMimeType("text/plain")){
                result = result + "\n" +
bodyPart.getContent().toString();
                break;
            }
        }
        email += result;
    }

    sentFolder.close(false);
}

catch (MessagingException e) {e.printStackTrace();}
catch (IOException e) { e.printStackTrace();}
return email;
}

public Message[] prefetchSent(String user,String password,int
start,int end) {
    Message[] sent = null;
    try {
        Properties properties = new Properties();
        properties.put("mail.imap.host",
"imap.gmail.com");

        properties.put("mail.imap.ssl.enable", "true");
        Session emailSession =
Session.getInstance(properties,
            new javax.mail.Authenticator() {

```



```

protected PasswordAuthentication
getPasswordAuthentication() {
    return new
    PasswordAuthentication(user,password);
    }
    });

    emailStore = (IMAPStore)
emailSession.getStore("imap");
    emailStore.connect(user, password);
    sentFolder =
emailStore.getFolder("[Gmail]").getFolder("Sent Mail");
    sentFolder.open(Folder.READ_ONLY);
    System.out.println(sentFolder.getMessageCount());
    sent =
sentFolder.getMessages(sentFolder.getMessageCount() -end +
1,sentFolder.getMessageCount() - start + 1);
    FetchProfile profile = new FetchProfile();
    profile.add(FetchProfileItem.ENVELOPE);
    profile.add(FetchProfileItem.FLAGS);
    profile.add(FetchProfileItem.CONTENT_INFO);
    profile.add("X-mailer");
    sentFolder.fetch(sent, profile);
    System.out.println("fetching done");
    sentFolder.close(false);

    }
    catch (NoSuchProviderException e)
{e.printStackTrace();}
    catch (MessagingException e) {e.printStackTrace();}
    catch (Exception e) {e.printStackTrace();}

    return sent;
}

```

Output:

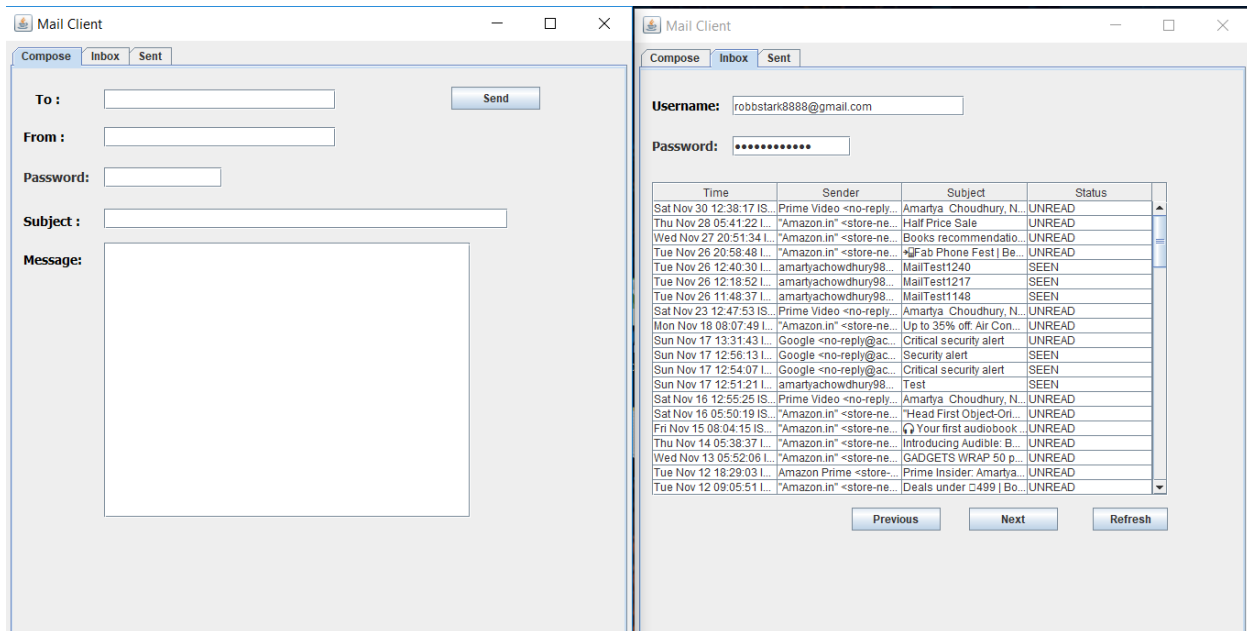


Fig 6: Reciever has no new messages

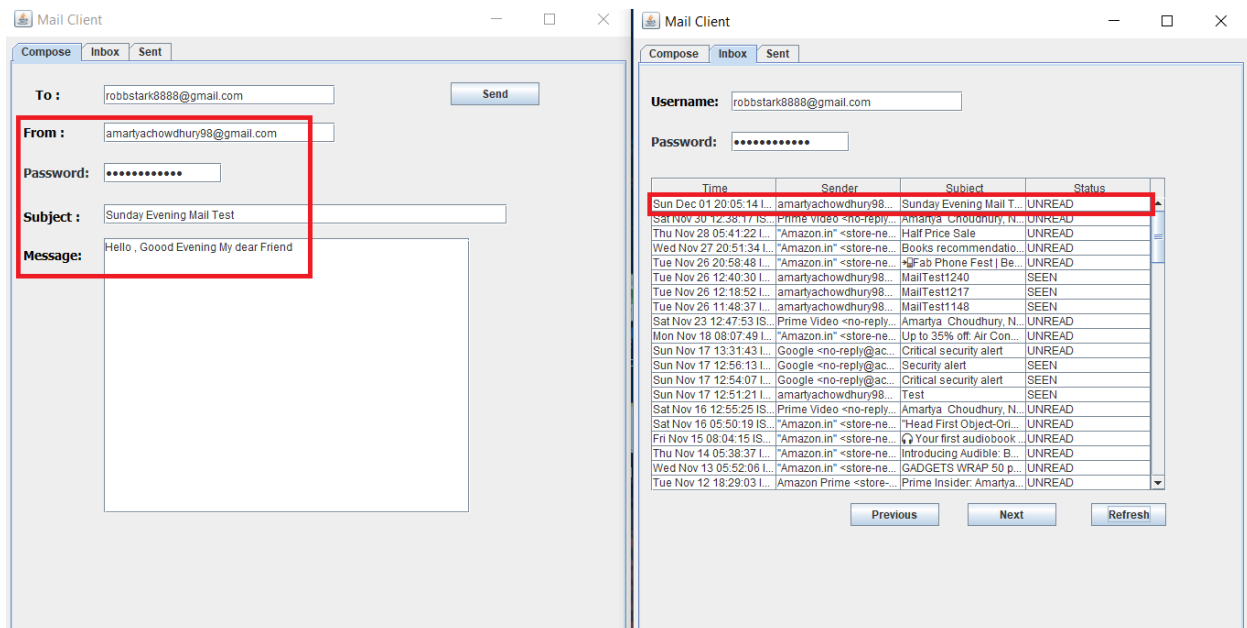


Fig 7: Sender's new message pops up in reciever's inbox

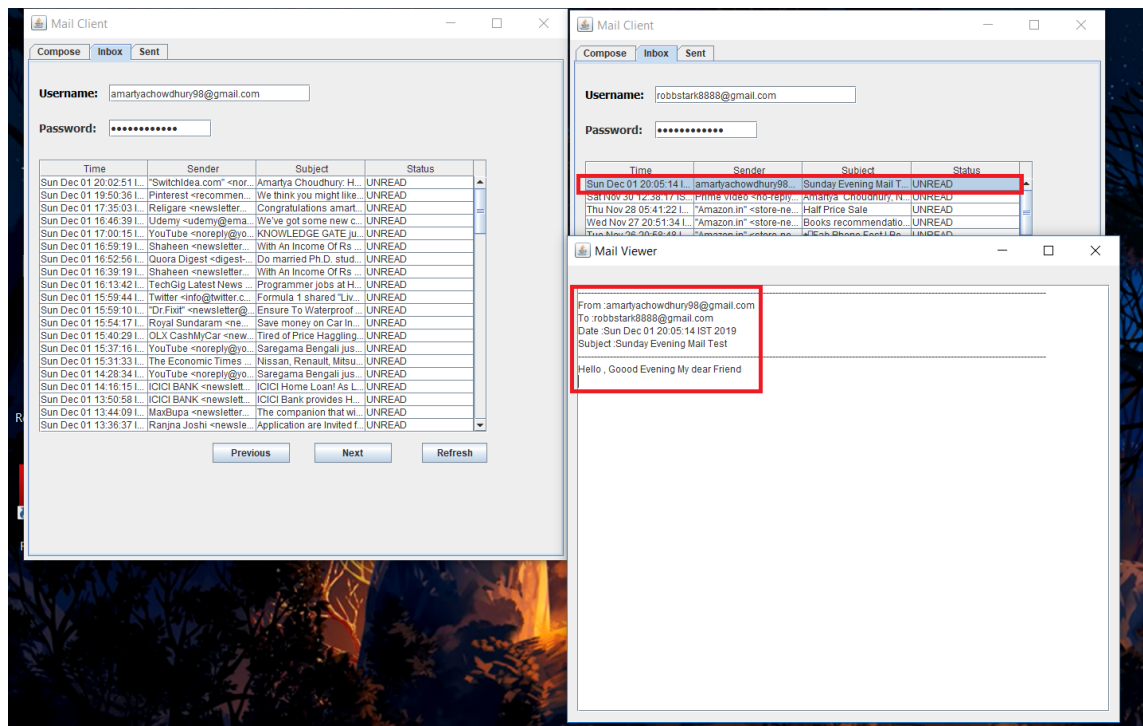


Fig 8: Mail Content fetched from inbox (Reciever Side)

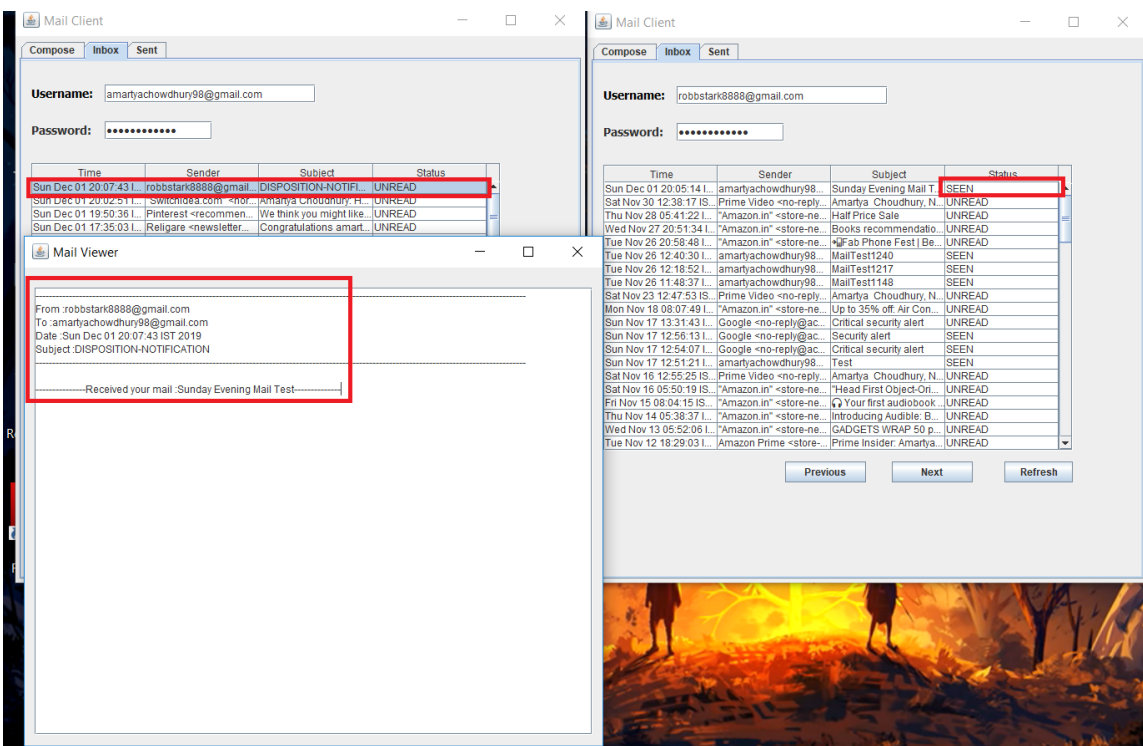


Fig 9: Mail flagged as SEEN(Reciever) and Read-Reciept delivered to Sender's inbox

References:

1. <https://whatismyipaddress.com/mail-server>
2. <https://www.quora.com/What-are-the-resources-to-learn-J2EE>
3. <http://liferayiseasy.blogspot.com/2015/09/mail-configuration-in-liferay-using.html>
4. <http://www.aosabook.org/en/index.html>
5. <https://gist.github.com/vasanthk/485d1c25737e8e72759fs>
6. <https://gist.github.com/vasanthk/485d1c25737e8e72759f>
7. <https://www.youtube.com/watch?v=x28ciavQ4ml>
8. https://www.tutorialspoint.com/javamail_api/javamail_api_folder_management.htm
9. <https://stackoverflow.com/questions/31981704/javamail-pull-messages-in-chunks-like-pagination-gmail-pop3>
10. <https://stackoverflow.com/questions/41165130/what-is-the-best-way-to-get-mails-using-javamail-api-fetch-or-getmessages>
11. <https://stackoverflow.com/questions/8322836/javamail-imap-over-ssl-quite-slow-bulk-fetching-multiple-messages?rq=1>
- 12.
13. <https://stackoverflow.com/questions/8322836/javamail-imap-over-ssl-quite-slow-bulk-fetching-multiple-messages?rq=1>
14. <https://stackoverflow.com/questions/2538481/javamail-performance>
- 15.
16. <https://stackoverflow.com/questions/8322836/javamail-imap-over-ssl-quite-slow-bulk-fetching-multiple-messages?rq=1>
- 17.
18. <https://stackoverflow.com/questions/7678919/javamail-mark-gmail-message-as-read>
19. <https://stackoverflow.com/questions/9263530/accessing-emails-from-gmail-using-imap-javamail-api>
20. <https://javaee.github.io/javamail/docs/api/javax/mail/Flags.Flag.html>
21. <https://docs.oracle.com/javase/tutorial/uiswing/components/table.html#show>
22. <https://stackoverflow.com/questions/17736870/setting-events-for-jtable-cells>
23. <https://stackoverflow.com/questions/11240368/how-to-read-text-inside-body-of-mail-using-javax-mail>
24. <https://stackoverflow.com/questions/40423574/read-sent-mails-using-smtp-java>
25. <https://www.rgagnon.com/javadetails/java-request-delivery-read-receipt-in-javamail.html>
26. https://www.chilkatsoft.com/p/p_313.asp
27. <https://www.ietf.org/rfc/rfc3464.txt>
28. <https://stackoverflow.com/questions/34157057/apache-common-email-how-to-get-email-delivery-notification-on-java-console>

29. <https://stackoverflow.com/questions/25600405/how-to-create-a-return-receipt-email-javamail>
 30. <https://serverfault.com/questions/949491/gmail-and-yahoo-not-accepting-dsn-option-from-rcpt-extended-commands>
 31. <https://tools.ietf.org/html/rfc1892>
 32. <http://edelstein.pebbles.cs.cmu.edu/jadeite/main.php?api=javamail&state=class&package=com.sun.mail.dsn&class=MultipartReport>
-