

BÁO CÁO THỰC HÀNH

Môn học: Lập trình mạng căn bản

Buổi báo cáo: Lab 05

Tên chủ đề: Sending & Receiving Email in C#

GVHD: Nguyễn Xuân Hà

Ngày thực hiện: 26/05/2024

THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT106.O23.2

STT	Họ và tên	MSSV	Email
1	Phạm Huỳnh Tấn Khang	22520624	22520624@gm.uit.edu.vn

1. ĐÁNH GIÁ KHÁC:

Nội dung	Kết quả
Tổng thời gian thực hiện bài thực hành trung bình	15 ngày
Link Video thực hiện (nếu có)	https://github.com/VitalsZen
Ý kiến (nếu có) + Khó khăn + Đề xuất ...	Khó khăn: <ul style="list-style-type: none">• Triển khai hệ thống mail gặp trục trặc ở khâu liên kết app-password• Mail pop3• Khó khăn ở việc gửi file HTML thông qua email và set background poster ở project số 5
Điểm tự đánh giá	10

Phần bên dưới của báo cáo này là báo cáo chi tiết của nhóm thực hiện.

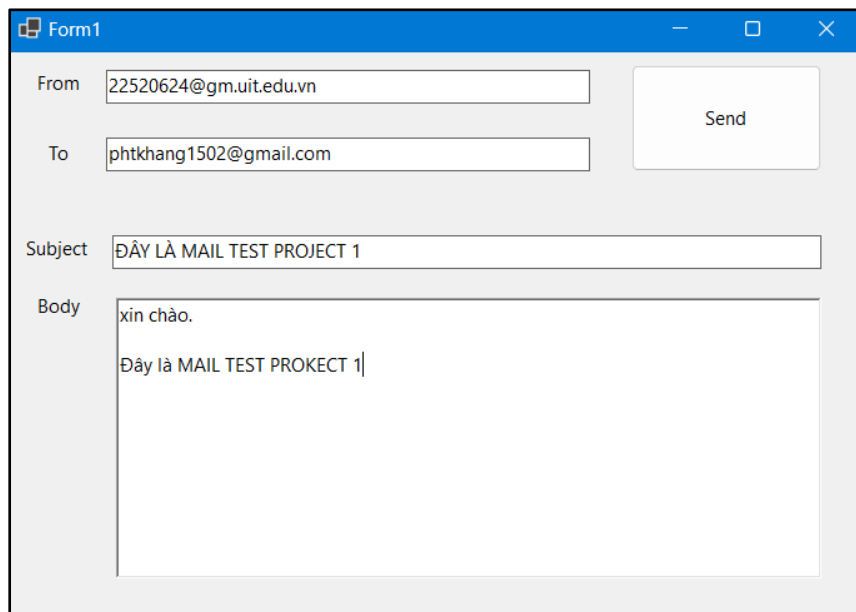
BÁO CÁO CHI TIẾT

MỤC LỤC

Bài 01: Viết ứng dụng cho phép gửi mail	3
Bài 02: Viết ứng dụng cho phép đọc mail (IMAP).....	4
Bài 03: Viết ứng dụng cho phép đọc mail (POP).	5
Bài 04: Quản lý phòng vé (phiên bản số 5).	6
Bài 05: Hôm nay ăn gì? (phiên bản số 5).....	17
Bài 06: Viết ứng dụng Email Client.....	22
YÊU CẦU CHUNG	30

Bài 01: Viết ứng dụng cho phép gửi mail

GIAO DIỆN



The screenshot shows a Windows application window titled "Form1". It contains four input fields and a button. The "From" field has the text "22520624@gm.uit.edu.vn". The "To" field has the text "phtkhang1502@gmail.com". The "Subject" field has the text "ĐÂY LÀ MAIL TEST PROJECT 1". The "Body" field is a text area containing the text "xin chào." and "Đây là MAIL TEST PROKECT 1". A "Send" button is located to the right of the "To" field.

Phương thức kích hoạt khi button “Send” được click: dùng để gửi mail thông qua Smtip Server với email, app – pass chỉ định

```
private void btConnect_Click(object sender, EventArgs e)
{
    string fromAddress = tbFrom.Text;
    string toAddress = tbTo.Text;
    string subject = tbSubject.Text;
    string body = rtbBody.Text;

    try
    {
        using (SmtpClient client = new SmtpClient("smtp.gmail.com", 587))
        {
            client.EnableSsl = true;

            // Kết nối an toàn
            string email = "tkhang1522004@gmail.com";
            string password = "yerm vvbv lvtv ncue";

            client.Credentials = new NetworkCredential(email, password);

            using (MailMessage mailMessage = new MailMessage(fromAddress, toAddress, subject, body))
            {
                client.Send(mailMessage);
            }
        }

        MessageBox.Show("Email gửi thành công");
    }
    catch (SmtpException ex)
    {
        MessageBox.Show($"SMTP lỗi: {ex.StatusCode} - {ex.Message}");
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Lỗi: {ex.Message}");
    }
}
```

Bài 02: Viết ứng dụng cho phép đọc mail (IMAP).

GIAO DIỆN

Email: phtkhang1502@gmail.com

Password: [masked]

Login

Total 4

	Subject	Mail From	Content	Time Received
▶	MAIL TEST MÔ HÌNH I...	"Phạm Huỳnh Tấ...	Xin chào đây là gmail để t...	08/06/2024 12:29:51 am
<input checked="" type="checkbox"/>	Chào Khang! Hãy h...	"The Google Acc...	Chào Khang! Bắt đầu sử d...	08/06/2024 12:32:10 am
	Don't let your job defin...	"LinkedIn" <noti...	Don't let your job define y...	08/06/2024 7:33:01 pm
	Cảnh báo bảo mật	"Google" <no-re...	[image: Google] Mật khẩu ...	09/06/2024 10:12:30 am
*				

Phương thức kích hoạt khi button “Login” được click: dùng để thiết lập kết nối IMAP tới email, app -pass chỉ định → triết xuất các mail bên trong email chỉ định và tải lên và đọc được trên datagridview

```
1 reference
private void btConnect_Click(object sender, EventArgs e)
{
    string email = tbEmail.Text;
    string password = tbPassword.Text;

    try
    {
        using (var client = new ImapClient())
        {
            client.Connect("imap.gmail.com", 993, true);
            client.Authenticate(email, password);

            var inbox = client.Inbox;
            inbox.Open(FolderAccess.ReadOnly);

            dtgvInfo.Rows.Clear();

            int limit = inbox.Count > 50 ? 50 : inbox.Count; // Giới hạn 50 mail gần nhất

            for (int i = 0; i < limit; i++)
            {
                var message = inbox.GetMessage(i);
                dtgvInfo.Rows.Add(message.Subject, message.From.ToString(), message.TextBody, message.Date.LocalDateTime);
            }

            lbCountTotal.Text = limit.ToString();

            client.Disconnect(true);
        }

        MessageBox.Show("Đăng nhập và triết xuất email thành công");
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Lỗi: {ex.Message}");
    }
}
```

Bài 03: Viết ứng dụng cho phép đọc mail (POP).

GIAO DIỆN

The screenshot shows a Windows application window titled "Form1". It has a login section with an "Email" field containing "phtkhang1502@gmail.com", a "Password" field with masked characters, and a "Login" button. Below the login section, it says "Total 4". Underneath is a table with 5 columns: "Subject", "Mail From", "Content", and "Time Received". The table contains 5 rows of email data. The first row is highlighted with a blue background. The second row has a checkbox checked. The third row has a truncated subject line. The fourth row has a truncated subject line. The fifth row has a truncated subject line.

	Subject	Mail From	Content	Time Received
▶	MAIL TEST MÔ HÌNH I...	"Phạm Huỳnh Tấ...	Xin chào đây là gmail để t...	08/06/2024 12:29:51 am
<input checked="" type="checkbox"/>	Chào Khang! Hãy h...	"The Google Acc...	Chào Khang! Bắt đầu sử d...	08/06/2024 12:32:10 am
	Don't let your job defin...	"LinkedIn" <noti...	Don't let your job define y...	08/06/2024 7:33:01 pm
	Cảnh báo bảo mật	"Google" <no-re...	[image: Google]Mật khẩu ...	09/06/2024 10:12:30 am
*				

Phương thức kích hoạt khi button “Login” được click: dùng để thiết lập kết nối POP tới email, app -pass chỉ định → triết xuất các mail bên trong email chỉ định và tải lên và đọc được trên datagridview

```
1 reference
private void btConnect_Click(object sender, EventArgs e)
{
    string hostname = "pop.gmail.com";
    int port = 995;
    string email = tbEmail.Text;
    string password = tbPassword.Text;

    try
    {
        using (var client = new Pop3Client())
        {
            client.Connect(hostname, port, true);
            client.Authenticate(email, password);

            int messageCount = client.Count;
            lbCountTotal.Text = messageCount.ToString(); // Hiển thị tổng số email

            dtgvInfo.Rows.Clear(); // Xóa dữ liệu cũ trong DataGridView

            // Lặp qua từng email và hiển thị thông tin
            for (int i = 0; i < messageCount; i++)
            {
                var message = client.GetMessage(i);
                var subject = message.Subject;

                // Lấy người gửi email
                var from = string.Join(", ", message.From.Select(f => ((MailboxAddress)f).Address));

                // Lấy nội dung email hoặc thông báo nếu không có văn bản thuần
                var body = message.TextBody ?? "[No Plain Text]";
                var dateReceived = message.Date.ToString();

                dtgvInfo.Rows.Add(subject, from, body, dateReceived);
            }

            client.Disconnect(true);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
    }
}
```

Bài 04: Quản lý phòng vé (phiên bản số 5).

GIAO DIỆN CỦA MAINFORM

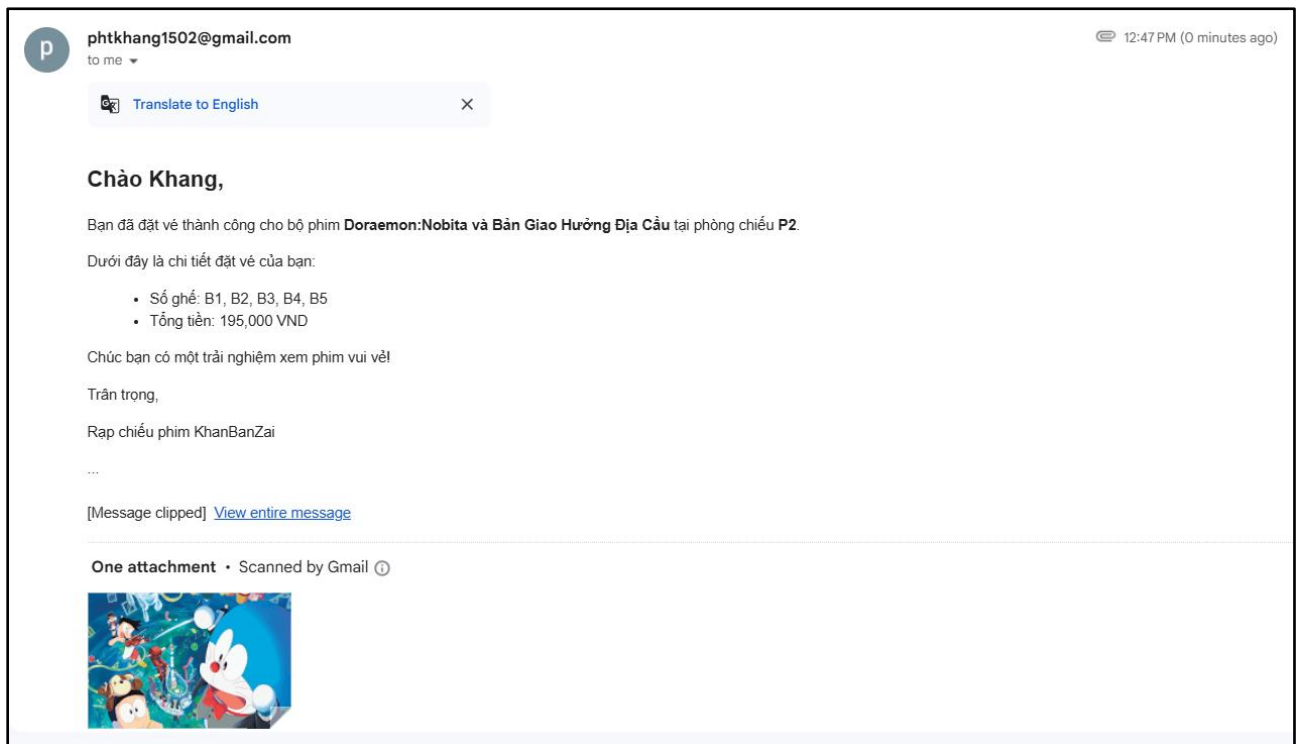
The MainForm window has a blue title bar with the text "MainForm". It contains a "Write a to File" button at the top left. Below it are input fields for "Số lượng phim" (number of movies), "Tên phim" (movie name), and "Phòng chiếu" (screening room) with radio buttons for P1, P2, and P3. There is also a "Giá vé chuẩn" (standard ticket price) input field with a placeholder "Nhập giá tiền (đơn vị: VND)". Below these are "Đường dẫn poster" (poster path) and "Số lần nhập còn lại" (remaining number of imports) input fields, followed by an "Add" button. At the bottom are "Thống kê" (statistics) and "Đặt vé" (book ticket) buttons. On the right side, there is a yellow text box with instructions: "Nhập thông tin của từng loại phim cho rạp(nếu muốn nhập mới) thì ấn vào 'Write to a file' - Hoặc ấn 'Đặt vé' để thực hiện đặt vé ngay và luôn vì file input5.txt đã có sẵn data về phim (không cần nhập lại data)".

GIAO DIỆN CỦA BOOKINGFORM

The Bai04_DatVe window has a title bar with the text "Bai04_DatVe". It contains a "Thông tin vé" (Ticket Information) section with input fields for "Họ tên:" (Name: Khang), "Email:" (22520624@gm.uit.edu.vn), "Phim:" (Doraemon:Nobita và Bản Giao I), and "Phòng:" (P2). To the right is a "Chọn chỗ ngồi" (Choose seat) section with a list of seats A1 through A5 and B1 through B5. B1, B2, B3, B4, and B5 are selected. Below the seating chart are "Xác nhận" (Confirm), "Xóa" (Delete), and "Thoát" (Exit) buttons. A "Warning !!" dialog box is open in the foreground, displaying: "Ho va ten: Khang", "Các vé đã chọn: B1 B2 B3 B4 B5", "Phòng chiếu: P2", and "Số tiền phải trả: 195000". The dialog has "Yes" and "No" buttons. In the background, a code editor shows a snippet of C# code:

```
MessageBox.Show(@"Nhập số nguyên và  
return;
```

GIAO DIỆN KHI NHẬN MAIL



PHÂN TÍCH VÀ GIẢI THÍCH

- Class CPhim (tổng hợp thông tin phòng vé) (SỬ DỤNG LẠI MÔ HÌNH CỦA LAB2)

```
20 references
internal class CPhim
{
    13 references
    public struct CPhong
    {
        9 references
        public string TenPhong { get; set; }
        7 references
        public string[] suat { get; set; }
        0 references
        public CPhong(string tenPhong, string[] suat)
        {
            TenPhong = tenPhong;
            this.suat = suat;
        }
        5 references
        public CPhong()
        {
            TenPhong = "";
            suat = new[] { "A1", "A2", "A3", "A4", "A5", "B1", "B2", "B3", "B4", "B5", "C1", "C2", "C3", "C4", "C5" };
        }
        0 references
        public string ImageUrl { get; set; }
    }
}
```

```

8 references
public string TenPhim { get; set; }
12 references
public List<CPhong> Phong { get; set; }
3 references
public long GiaVe { get; set; }
5 references
public long TongTien { get; set; }
3 references
public int Rank { get; set; }
4 references
public string ImageUrl { get; set; }
2 references
public CPhim()
{
    TenPhim = string.Empty;
    Phong = new List<CPhong>();
    GiaVe = 0;
    TongTien = 0;
    Rank = -1;
    ImageUrl = string.Empty;
}
0 references
~CPhim()
{
}

```

PHÂN TẠO THÔNG TIN PHIM

- Hàm kích hoạt khi được click của button "Write to a file"
 - Kiểm tra xem trường "Total" có được nhập hay không. Nếu không, hiển thị thông báo yêu cầu nhập số lượng phim

```

1 reference
private void bt1_WtF_Click(object sender, EventArgs e)
{
    if (tbTotal.Text == "")
    {
        MessageBox.Show("Nhập số lượng phim vào Total");
        tbTotal.Enabled = true;
        return;
    }
}

```

- Hàm 'EnableInput()': Kích hoạt các ô nhập liệu để người dùng có thể nhập thông tin về phim

```

1 reference
private void EnableInput()
{
    tbName.Enabled = true;
    cbRoom1.Enabled = true;
    cbRoom2.Enabled = true;
    cbRoom3.Enabled = true;
    tbCost.Enabled = true;
    btAdd.Enabled = true;
}

```


- Hàm kích hoạt khi có chuỗi dữ liệu hiển thị bị thay đổi ở textbox trường "Total"
 - Đảm bảo chỉ nhập số nguyên
 - Kiểm tra xem giá trị nhập vào có phải là số nguyên hay không và hiển thị thông báo nếu không phải

```
private void tbTotal_TextChanged(object sender, EventArgs e)
{
    if (!Int32.TryParse(tbTotal.Text, out int temp))
    {
        MessageBox.Show(@"Nhập số nguyên vào thành "So lương phim": ");
        return;
    }
    n = temp;
    tbCount.Text = n.ToString();
    MessageBox.Show("Nhap day du thong tin phim vao cac o phia duoi!! ");
    EnableInput();
}
```

- Hàm kích hoạt khi được click của button "Add": Thêm thông tin của phim vào danh sách sau khi kiểm tra thông tin nhập vào và đảm bảo là thông tin hợp lệ

```

private void btAdd_Click(object sender, EventArgs e)
{
    try
    {
        CPhim Phim = new CPhim();
        int num = Int32.Parse(tbCount.Text);
        Phim.TenPhim = tbName.Text;
        if (cbRoom1.CheckState == CheckState.Checked)
        {
            Phim.Phong.Add(new CPhim.CPhong { TenPhong = cbRoom1.Text.Trim() });
        }
        if (cbRoom2.CheckState == CheckState.Checked)
        {
            Phim.Phong.Add(new CPhim.CPhong { TenPhong = cbRoom2.Text.Trim() });
        }
        if (cbRoom3.CheckState == CheckState.Checked)
        {
            Phim.Phong.Add(new CPhim.CPhong { TenPhong = cbRoom3.Text.Trim() });
        }
        Phim.GiaVe = Int32.Parse(tbCost.Text);
        tbCount.Text = (num - 1).ToString();

        Phims.Add(Phim);
        MessageBox.Show("Nhập thành công!!, con lai " + tbCount.Text + " lan nhập");
        tbName.Text = "";
        tbCost.Text = "";
        cbRoom1.CheckState = CheckState.Unchecked;
        cbRoom2.CheckState = CheckState.Unchecked;
        cbRoom3.CheckState = CheckState.Unchecked;
        tbTotal.Enabled = false;

        if (tbCount.Text == "0")
        {
            MessageBox.Show("Thực hiện đầy thông tin vào file ....", "Cảnh báo", MessageBoxButtons.OK);
            SerializeJson(Phims, "input5.txt");
            SerializeJson(Phims, "output5.txt");
            //DisableInput();
            //bt3_Read.Enabled = true;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

```

- Hàm 'DeserializeJson(string Filepath)': Đọc và chuyển đổi chuỗi JSON từ tệp tin thành danh sách các đối tượng phim
- Hàm 'SerializeJson(object obj, string Filepath)': Chuyển đổi danh sách các đối tượng phim thành chuỗi JSON và lưu vào tệp tin tại đường dẫn được chỉ định

```

private List<CPhim> DeserializeJson(string Filepath)
{
    string json = File.ReadAllText(Filepath);
    List<CPhim> Phims = JsonSerializer.Deserialize<List<CPhim>>(json);
    return Phims;
}

2 references
private void SerializeJson(object obj, string Filepath)
{
    string json = JsonSerializer.Serialize(obj);
    File.WriteAllText(Filepath, json);
}

```

- Hàm kích hoạt khi được click của button "Đặt vé". Mở form mới để thực hiện quá trình đặt vé cho các phim

```
private void btDatVe_Click(object sender, EventArgs e)
{
    var Form = new Bai05_DatVe();
    Form.Show();
}
```

- Hàm 'RankPhim(List<CPhim> cphimList)': Sắp xếp danh sách phim theo thứ tự giảm dần của tổng doanh thu và gán xếp hạng cho mỗi phim

```
private static void RankPhim(List<CPhim> cphimList)
{
    for (int i = 0; i < cphimList.Count - 1; i++)
    {
        for (int j = 0; j < cphimList.Count - 1 - i; j++)
        {
            if (cphimList[j].TongTien < cphimList[j + 1].TongTien)
            {
                CPhim temp = cphimList[j];
                cphimList[j] = cphimList[j + 1];
                cphimList[j + 1] = temp;
            }
        }
    }

    for (int i = 0; i < cphimList.Count; i++)
    {
        cphimList[i].Rank = i + 1;
    }
}
```

- Hàm kích hoạt khi được click của button "Read"
 - Đọc thông tin từ tệp tin và hiển thị ra giao diện sau khi sắp xếp và tính toán các thông tin liên quan đến doanh thu và xếp hạng của mỗi phim
 - Chèn vào **ProgressBar** progressbar1 để cho người dùng theo dõi quá trình đọc file

```

private void bt3_Read_Click(object sender, EventArgs e)
{
    progressBar1.Value = 0;
    try
    {
        Phims = DeserializeJson("output5.txt");
        RankPhim(Phims);
        rtbl_Show.Clear();
        int progressStep = 100 / Phims.Count;
        foreach (var c in Phims)
        {
            progressBar1.Value += progressStep;
            progressBar1.Refresh();

            rtbl_Show.Text += "Ten phim: " + c.TenPhim + '\n';
            double vetong = 0;
            double veban = 0;
            foreach (CPhim.CPhong c2 in c.Phong)
            {
                vetong += 15;
                veban += c2.suat.Length;
            }
            rtbl_Show.Text += "So ve ban duoc: " + (vetong - veban).ToString() + '\n' + "So ve ton: " + veban.ToString() + '\n';
            rtbl_Show.Text += "Ti le ve ban duoc: " + (((vetong - veban) / vetong) * 100).ToString("0.00") + "%" + '\n';
        }
    }
    catch (JsonException)
    {
        MessageBox.Show("File thông kê (output5.txt) hiện không có dữ liệu để trích xuất hoặc lỗi về Json ");
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

```

PHẦN ĐẶT VÉ

Hàm kích hoạt khi có sự thay đổi về dữ liệu được chọn tại combobox cb_Movie2. Nó sẽ điền các rạp chiếu có sẵn vào cb1_Theater cho phim đã chọn.

```

1 reference
private void cb2_Movie_SelectionChangeCommitted(object sender, EventArgs e)
{
    cb1_Theater.Items.Clear();
    if (cb2_Movie.SelectedItem != null)
    {
        foreach (CPhim Phim in Phims)
        {
            if (Phim.TenPhim == cb2_Movie.SelectedItem.ToString())
            {
                //MessageBox.Show("1");
                SelectedPhim = Phim;
                break;
            }
        }

        if (SelectedPhim != null && SelectedPhim.Phong != null)
        {
            //MessageBox.Show("2");
            foreach (CPhim.CPhong phong in SelectedPhim.Phong)
            {
                cb1_Theater.Items.Add(phong.TenPhong);
            }
        }
        cb1_Theater.Enabled = true;
    }
}

```

Hàm kích hoạt khi có sự thay đổi về dữ liệu được chọn tại combobox cb1_Theater. Nó sẽ điền danh sách các ghế ngồi có sẵn vào CheckedListBox clb1_Seats cho rạp đã chọn.

```

1 reference
private void cb1_Theater_SelectionChangeCommitted(object sender, EventArgs e)
{
    clb1_Seats.Items.Clear();
    if (cb1_Theater.SelectedItem != null && SelectedPhim != null && SelectedPhim.Phong != null)
    {
        //MessageBox.Show("3"); ;
        foreach (CPhim.CPhong c in SelectedPhim.Phong)
        {
            if (c.TenPhong == cb1_Theater.SelectedItem.ToString())
            {
                //MessageBox.Show("4"); ;
                SelectedPhong = c;
                break;
            }
        }

        if (SelectedPhong.suat != null)
        {
            clb1_Seats.Items.AddRange(SelectedPhong.suat);
        }
    }
}

```

Hàm kích hoạt khi có một Item được tick ở trong clb1_Seats. Nó sẽ bật hoặc tắt nút bt1_Cofrm dựa trên số lượng ghế được chọn → nhằm cải thiện UI và khóa trường hợp không tick nhưng vẫn đặt vé

```
1 reference
private void clb1_Seats_ItemCheck(object sender, ItemCheckEventArgs e)
{
    if (clb1_Seats.CheckedItems.Count == 1)
    {
        if (e.NewValue == CheckState.Unchecked)
            bt1_Confirm.Enabled = false;
        else
            bt1_Confirm.Enabled = true;
    }
}
```

Hàm kích hoạt khi có button “Xác nhận” được click. Nó tính toán tổng tiền, xác nhận đặt vé, cập nhật ghế còn trống và gửi email xác nhận cho người đặt


```

1 reference
private void bt1_Confirm_Click(object sender, EventArgs e)
{
    long tong = 0; // tính tổng tiền từ checked seats
    long cost = SelectedPhim.GiaVe;
    foreach (string c in clb1_Seats.CheckedItems)
    {
        if (new[] { "A1", "A5", "B1", "B5", "C1", "C5" }.Contains(c))
        {
            tong += cost * 1 / 4;
        }
        else if (new[] { "A2", "A3", "A4", "C2", "C3", "C4" }.Contains(c))
        {
            tong += cost * 1;
        }
        else
        {
            tong += cost * 2;
        }
    }

    string s = "Họ và tên: " + tb1_Name.Text;
    s += System.Environment.NewLine + "Các vé đã chọn: ";
    foreach (string c in clb1_Seats.CheckedItems)
    {
        s += c + " ";
    }
    s += System.Environment.NewLine;
    s += "Phòng chiếu: " + cb1_Theater.Text;
    s += System.Environment.NewLine;
    s += "Số tiền phải trả: " + tong.ToString();

    if (MessageBox.Show(s, "Warning !!", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.No) // thông báo lần cuối chắc chắn hay ko ?
    {
        return;
    }
    else
    {
        MessageBox.Show("Bạn đã đặt vé thành công.", "Congratulations", MessageBoxButtons.OK, MessageBoxIcon.Information);
        SelectedPhim.TongTien += tong;
        HashSet<string> temp = SelectedPhong.suat.ToHashSet();
        foreach (string c in clb1_Seats.CheckedItems)
        {
            temp.Remove(c);
        }
        SelectedPhong.suat = temp.ToArray();
    }
}

```

```

foreach (var c in SelectedPhim.Phong)
{
    if (c.TenPhong == SelectedPhong.TenPhong)
    {
        SelectedPhim.Phong.Remove(c);
        SelectedPhim.Phong.Add(SelectedPhong);
        break;
    }
}

foreach (var c in Phims)
{
    if (SelectedPhim.TenPhim == c.TenPhim)
    {
        Phims.Remove(c);
        Phims.Add(SelectedPhim);
        break;
    }
}

SerializeJson(Phims, "output5.txt");

SendConfirmationEmail(tong);
// blank các ô -> đẹp
tb1_Name.Text = "";
cb2_Movie.Text = "";
cb1_Theater.Text = "";
cb1_Theater.Enabled = false;
SelectedPhim = null;
SelectedPhong = new CPhim.CPhong();
cb1_Theater.Items.Clear();
clb1_Seats.Items.Clear();
}
}

```

Hàm `SendConfirmationEmail(long)` sử dụng để gửi mail xác nhận đến mail người đặt

```
1 reference
private void SendConfirmationEmail(long totalCost)
{
    string base64Image = ConvertImageToBase64($"{SelectedPhim.ImageUrl}"); // Chuyển đổi hình ảnh thành chuỗi base64
    string emailContent = @"
    <html>
    <head>
    <style>
    body {
        background-image: url('data:image/png;base64,{base64Image}');
        background-size: cover;
        background-repeat: no-repeat;
        font-family: Arial, sans-serif;
        color: black;
    }
    .container {
        max-width: 600px;
        margin: auto;
        padding: 20px;
    }
    </style>
    </head>
    <body>
    <div class='container'>
    <h2>Chào {tb1_Name.Text},</h2>
    <p>Bạn đã đặt vé thành công cho bộ phim <strong>{SelectedPhim.TenPhim}</strong> tại phòng chiếu <strong>{cb1_Theater.Text}</strong>.</p>
    <p>Dưới đây là chi tiết đặt vé của bạn:</p>
    <ul>
    <li>Số ghế: {string.Join(" ", clb1_Seats.CheckedItems.Cast<string>())}</li>
    <li>Tổng tiền: {totalCost.ToString("N0")} VND</li>
    </ul>
    <p>Chúc bạn có một trải nghiệm xem phim vui vẻ!</p>
    <p>Trân trọng,</p>
    <p>Rap chiếu phim KhanBanZai</p>
    </div>
    </body>
    </html>";

    string fromAddress = "phtkhang1502@gmail.com";
    string toAddress = tb2_Mail.Text;
    string subject = "Booking Info";
    string body = emailContent;
}
```

```
try
{
    using (SmtpClient client = new SmtpClient("smtp.gmail.com", 587))
    {
        client.EnableSsl = true; // Kích hoạt SSL

        string email = "phtkhang1502@gmail.com";
        string password = "apfs cfto uvbx pefu";

        client.Credentials = new NetworkCredential(email, password); // Đặt thông tin xác thực

        using (MailMessage mailMessage = new MailMessage())
        {
            mailMessage.From = new MailAddress(fromAddress);
            mailMessage.To.Add(toAddress);
            mailMessage.Subject = subject;
            mailMessage.Body = body;
            mailMessage.IsBodyHtml = true; // Đặt nội dung email là HTML

            Attachment attachment = new Attachment($"{SelectedPhim.ImageUrl}"); // Đính kèm hình ảnh
            attachment.ContentDisposition.Inline = true;
            attachment.ContentDisposition.DispositionType = DispositionTypeNames.Inline;
            mailMessage.Attachments.Add(attachment);

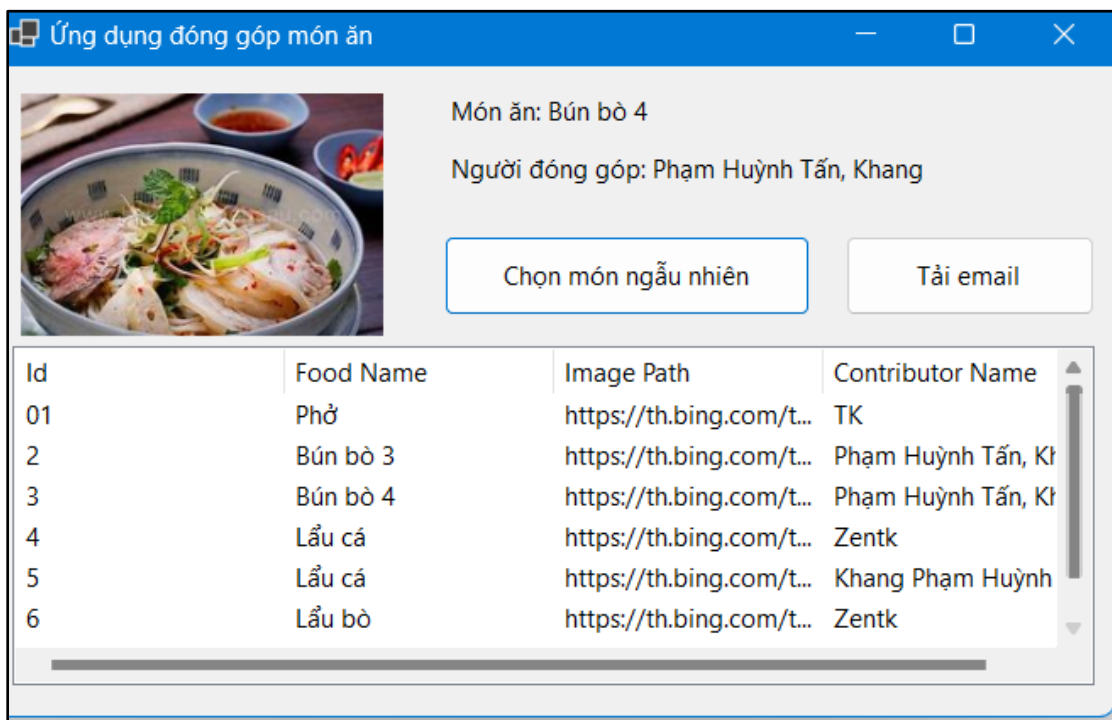
            client.Send(mailMessage); // Gửi email
        }
    }
}
catch (SmtpException ex)
{
    MessageBox.Show($"SMTP error: {ex.StatusCode} - {ex.Message}"); // Hiển thị lỗi SMTP
}
catch (Exception ex)
{
    MessageBox.Show($"Error: {ex.Message}"); // Hiển thị lỗi chung
}
```


Hàm `ConvertImageToBase64(string)` dùng để chuyển đổi đường dẫn hình ảnh thành chuỗi base64

```
1 reference
private string ConvertImageToBase64(string imagePath)
{
    byte[] imageArray = File.ReadAllBytes(imagePath);
    return Convert.ToBase64String(imageArray);
}
1 reference
```

Bài 05: Hôm nay ăn gì? (phiên bản số 5)

GIAO DIỆN



Hàm `InitializeDatabase()` dùng để thiết lập database `Food.db` để lưu trữ thông tin món ăn (nếu cơ sở dữ liệu `Food.db` không có trong đường dẫn chỉ định)

Hàm kích hoạt khi nút “Tải Email” được click

Thực thi hàm `FetchEmails()`: Triết xuất mail từ email chỉ định

Thực thi hàm `LoadDataToListView()`: tải thông tin triết xuất lên `ListView`

```

1 reference
private void InitializeDatabase()
{
    if (!File.Exists("Food.db")) // Kiểm tra xem tệp cơ sở dữ liệu đã tồn tại chưa
    {
        SQLiteConnection.CreateFile("Food.db"); // Tạo tệp cơ sở dữ liệu nếu chưa tồn tại
        using (var conn = new SQLiteConnection(dbPath))
        {
            conn.Open();
            string sql = @"
                CREATE TABLE FOOD (
                    ID TEXT PRIMARY KEY UNIQUE NOT NULL,
                    FOODNAME TEXT NOT NULL,
                    IMAGEPATH TEXT NOT NULL,
                    CONTRIBUTOR TEXT NOT NULL
                );";
            SQLiteCommand command = new SQLiteCommand(sql, conn);
            command.ExecuteNonQuery(); // Tạo bảng FOOD
        }
    }
}

1 reference
private void btnFetchEmails_Click(object sender, EventArgs e)
{
    FetchEmails(); // Gọi hàm FetchEmails để lấy email
    LoadDataToListView(); // Gọi hàm LoadDataToListView để tải dữ liệu vào ListView
}

```

- Hàm FetchEmails() dùng để thiết lập kết IMAP tới Mail chỉ định
 - Triết xuất mail từ trong Tab Inbox.
 - Check và ghi nhận những mail có tiêu đề “Đóng góp món ăn” và trạng thái chưa được mở vào trong biến **uids** (trạng thái mail có màu in đậm khi được nhận trên web Gmail)
 - Chạy vòng lặp, đọc và điều chế mail thông qua hàm **ProcessEmail(message)** cho tất cả mail trong **uids**
 - Cấm cờ “Đã đọc” cho tất cả mail được điều chế (nhằm tránh việc lấy mail trên thêm vào cơ sở dữ liệu sau này)

```

1 reference
private void FetchEmails()
{
    using (var client = new ImapClient())
    {
        failed = "";
        client.Connect("imap.gmail.com", 993, true); // Kết nối đến máy chủ IMAP
        client.Authenticate("phtkhang1502@gmail.com", "tsagjqjssgkrsnkn"); // Xác thực người dùng

        client.Inbox.Open(FolderAccess.ReadWrite); // Mở hộp thư đến
        var query = SearchQuery.SubjectContains("Đóng góp món ăn").And(SearchQuery.NotSeen); // Tìm kiếm email chưa đọc với tiêu đề chứa "Đóng góp món ăn"
        var uids = client.Inbox.Search(query); // Lấy danh sách UID của email phù hợp

        foreach (var uid in uids)
        {
            var message = client.Inbox.GetMessage(uid); // Lấy nội dung email
            ProcessEmail(message); // Xử lý email
            client.Inbox.AddFlags(uid, MessageFlags.Seen, true); // Đánh dấu email là đã đọc
        }

        client.Disconnect(true); // Ngắt kết nối
    }
}

```

Hàm SaveContribution(string,string,string) dùng để lưu phần dữ liệu đã điều chế vào cơ sở dữ liệu

```
1 reference
private void SaveContribution(string foodName, string imagePath, string contributorName)
{
    using (var conn = new SQLiteConnection(dbPath))
    {
        conn.Open();
        string countSql = "SELECT COUNT(*) FROM FOOD";
        using (var countCmd = new SQLiteCommand(countSql, conn))
        {
            Count = Convert.ToInt32(countCmd.ExecuteScalar()); // Đếm số lượng bản ghi hiện có
        }

        if (contributorName == "")
        {
            contributorName = "Người ẩn danh"; // Nếu không có tên người đóng góp, gán giá trị mặc định
        }
        string sql = "INSERT INTO FOOD (ID, FOODNAME, IMAGEPATH, CONTRIBUTOR) VALUES (@ID, @FOODNAME, @IMAGEPATH, @CONTRIBUTOR)";
        using (var cmd = new SQLiteCommand(sql, conn))
        {
            cmd.Parameters.AddWithValue("@ID", Count + 1); // Gán ID dựa trên số lượng bản ghi
            cmd.Parameters.AddWithValue("@FOODNAME", foodName);
            cmd.Parameters.AddWithValue("@IMAGEPATH", imagePath);
            cmd.Parameters.AddWithValue("@CONTRIBUTOR", contributorName);
            cmd.ExecuteNonQuery(); // Thêm bản ghi mới vào bảng FOOD
        }
    }
}
```

- Phương thức kích hoạt khi button “Chọn món ngẫu nhiên” được click: chọn ngẫu nhiên món ăn trong cơ sở dữ liệu Food.db thông qua hàm **DisplayRandomFood()**
- Hàm DisplayRandomFood(): thể hiện ra món ăn đã chọn ngẫu nhiên từ cơ sở dữ liệu
 - Thiết lập kết nối với cơ sở dữ liệu, truyền câu truy vấn lấy một bộ dữ liệu ngẫu nhiên trong Table FOOD
 - Thể hiện thông tin của bộ dữ liệu được chọn tại UI

```
private void btnRandomFood_Click(object sender, EventArgs e)
{
    DisplayRandomFood(); // Gọi hàm DisplayRandomFood để hiển thị món ăn ngẫu nhiên
}

1 reference
private void DisplayRandomFood()
{
    using (var conn = new SQLiteConnection(dbPath))
    {
        conn.Open();
        string sql = "SELECT * FROM FOOD ORDER BY RANDOM() LIMIT 1"; // Lấy một món ăn ngẫu nhiên
        using (var cmd = new SQLiteCommand(sql, conn))
        {
            using (var reader = cmd.ExecuteReader())
            {
                if (reader.Read())
                {
                    string foodName = reader["FOODNAME"].ToString();
                    string imagePath = reader["IMAGEPATH"].ToString();
                    string contributorName = reader["CONTRIBUTOR"].ToString();

                    lblFoodName.Text = $"Món ăn: {foodName}";
                    lblContributorName.Text = $"Người đóng góp: {contributorName}";

                    LoadImageFromUrl(imagePath); // Gọi hàm LoadImageFromUrl để tải hình ảnh món ăn
                }
            }
        }
    }
}
```

Hàm LoadImageFromURL(string) dùng để tải hình ảnh dạng URL lên pictureBox

```
1 reference
private void LoadImageFromUrl(string url)
{
    try
    {
        var request = WebRequest.Create(url);
        using (var response = request.GetResponse())
        using (var stream = response.GetResponseStream())
        {
            pictureBoxFood.Image = System.Drawing.Image.FromStream(stream); // Tải và hiển thị hình ảnh từ URL
        }
    }
    catch
    {
        pictureBoxFood.Image = null; // Nếu lỗi, đặt hình ảnh thành null
    }
}
```

Hàm LoadDataToListView() dùng để lấy dữ liệu trong cơ sở dữ liệu và tải lên listview

```
1 reference
private void LoadDataToListView()
{
    listViewContributions.Items.Clear(); // Xóa các mục hiện có trong ListView

    using (var conn = new SQLiteConnection(dbPath))
    {
        conn.Open();
        string sql = "SELECT * FROM FOOD";
        using (var cmd = new SQLiteCommand(sql, conn))
        {
            using (var reader = cmd.ExecuteReader())
            {
                while (reader.Read())
                {
                    var item = new ListViewItem(reader["ID"].ToString());
                    item.SubItems.Add(reader["FOODNAME"].ToString());
                    item.SubItems.Add(reader["IMAGEPATH"].ToString());
                    item.SubItems.Add(reader["CONTRIBUTOR"].ToString());
                    listViewContributions.Items.Add(item); // Thêm mục vào ListView
                }
            }
        }
    }
}
```

- Hàm ProcessEmail(MimeMessage) dùng để điều chế mail được đưa vào
 - Xác định tên người gửi (nếu không có thì đổi sang “Người ẩn danh”)
 - Xác định thân của mail
 - Dùng hàm Split để chia nội dung dạng text của mail sang dữ liệu cần truyền vào cơ sở dữ liệu với cấu trúc mail có dạng: với mỗi
 - <Tên món ăn 1>;<Hình ảnh 1>
 - <Tên món ăn n>;<Hình ảnh n>
 - Text Visualizer: “<Tên món ăn 1>;/r/n<Hình ảnh 1>/r/n <Tên món ăn 2>
 - Đưa dữ liệu đã điều chế thành tham số đầu vào của hàm SaveContributon(string,string,string)

- Đặt biến failed nhằm check lỗi trùng tên và các món ăn trùng tên sẽ không được thêm vào cơ sở dữ liệu

```
1 reference
private void ProcessEmail(MimeMessage message)
{
    string contributorName = message.From.Mailboxes.FirstOrDefault()?.Name ?? "Người ẩn danh";
    // Lấy tên người đóng góp hoặc gán giá trị mặc định
    string body = message.TextBody;
    body = body.Replace("\r\n", "");
    var lines = body.Split("\r\n");
    foreach (var line in lines)
    {
        var parts = line.Split(';');
        if (parts.Length == 2)
        {
            var foodName = parts[0];
            var imagePath = parts[1];
            // Kiểm tra xem món ăn đã tồn tại trong cơ sở dữ liệu chưa trước khi thêm vào
            if (!FoodExists(foodName))
            {
                SaveContribution(foodName, imagePath, contributorName); // Lưu đóng góp món ăn nếu chưa tồn tại
            }
            else
            {
                failed += foodName + "; ";
            }
        }
    }
    if (failed != "")
        MessageBox.Show("Món ăn thêm vào thất bại: " + failed + "\nLỗi: Trùng tên với món ăn đã có trong cơ sở dữ liệu", "Warning!!");
}
```

Hàm FoodExists(string) dùng để check tên món ăn trong cơ sở dữ liệu (Case Ignored)
 Nếu trùng tên truyền return true;
 Nếu không return false;

```
1 reference
private bool FoodExists(string foodName)
{
    bool exists = false;
    using (var conn = new SQLiteConnection(dbPath))
    {
        conn.Open();
        string sql = "SELECT COUNT(*) FROM FOOD WHERE FOODNAME = @FOODNAME"; // Kiểm tra xem món ăn đã tồn tại chưa
        using (var cmd = new SQLiteCommand(sql, conn))
        {
            cmd.Parameters.AddWithValue("@FOODNAME", foodName);
            int count = Convert.ToInt32(cmd.ExecuteScalar());
            exists = count > 0; // Nếu số lượng lớn hơn 0, món ăn đã tồn tại
        }
    }
    return exists;
}
```

Bài 06: Viết ứng dụng Email Client.

GIAO DIỆN MENUMAIL.CS

The screenshot shows the 'Email Client' window. It has a blue title bar and a light gray background. The window is divided into two main sections: 'Login Info' and 'Setting'.

Login Info:

- Email: phtkhang1502@gmail.com
- Pass: *****

Setting:

- IMAP: imap.gmail.com (Port: 993)
- SMTP: smtp.gmail.com (Port: 587)

Below the settings, there is a table of received emails:

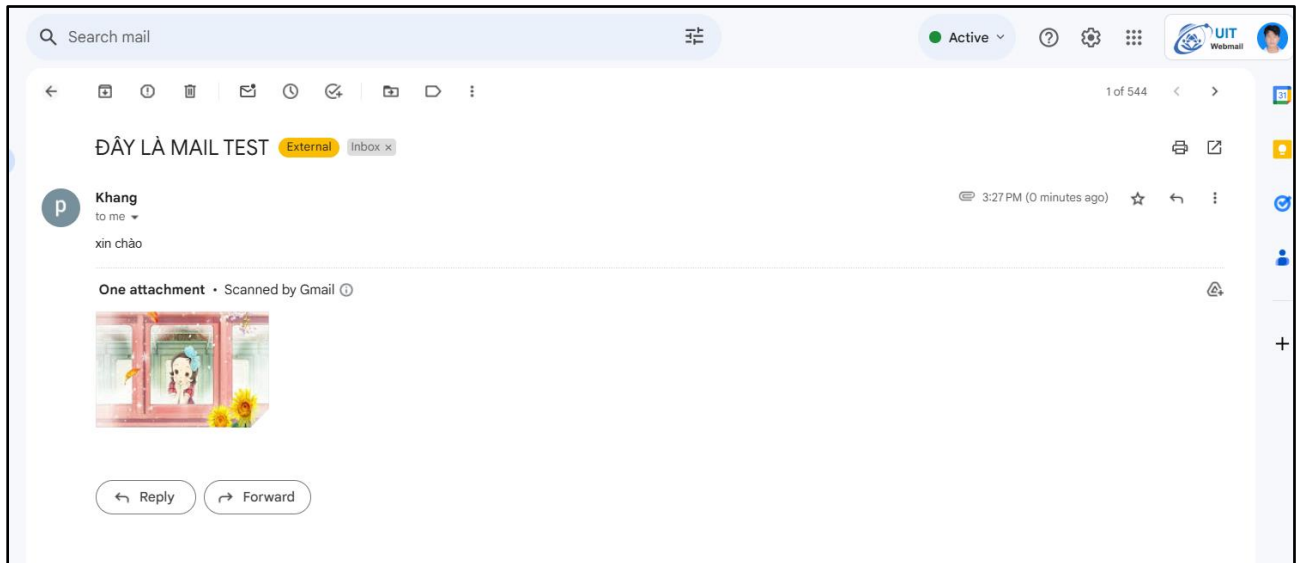
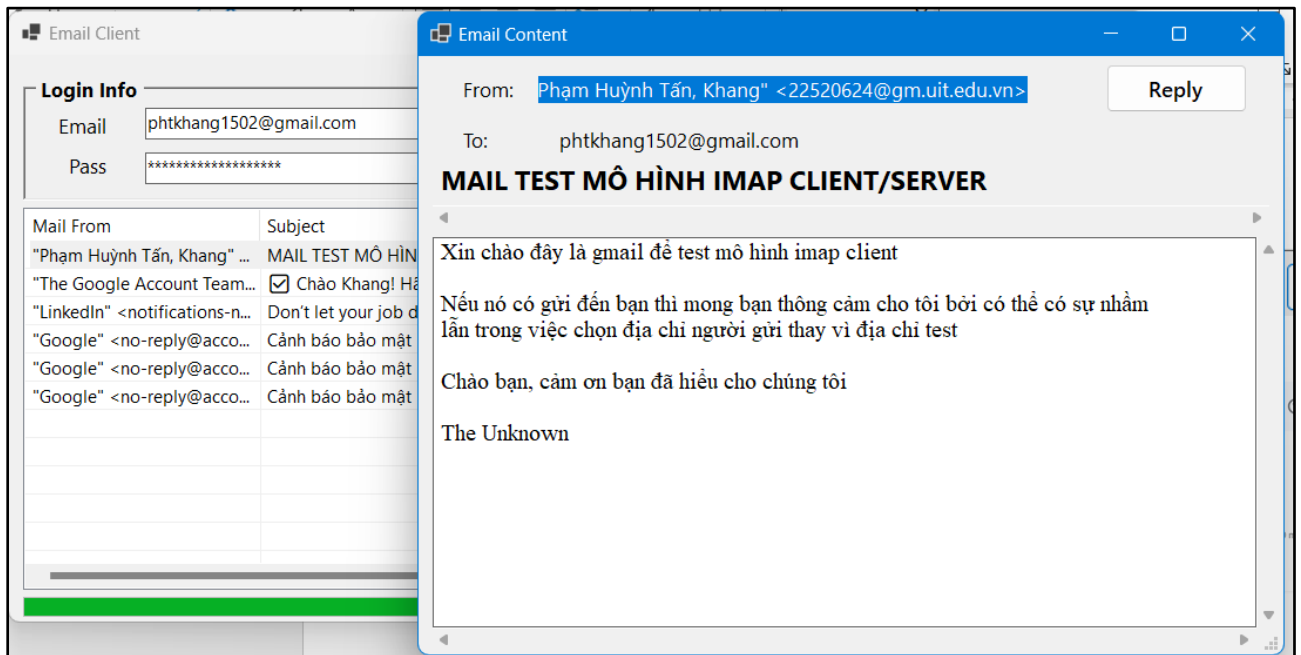
Mail From	Subject	Content	Date Received
"Phạm Huỳnh Tấn, Khang" ...	MAIL TEST MÔ HÌNH IMAP...	Xin chào đây là gmail để test mô hình ima...	08/06/2024 12:29...
"The Google Account Team..."	<input checked="" type="checkbox"/> Chào Khang! Hãy hoàn...	Chào Khang! Bắt đầu sử dụng Google trên...	07/06/2024 10:32...
"LinkedIn" <notifications-n...	Don't let your job define y...	Don't let your job define you. Here's how t...	08/06/2024 12:33...
"Google" <no-reply@acco...	Cảnh báo bảo mật	[image: Google] Mật khẩu ứng dụng đã đ...	09/06/2024 3:12...
"Google" <no-reply@acco...	Cảnh báo bảo mật	[image: Google] Mật khẩu ứng dụng được...	09/06/2024 5:35...
"Google" <no-reply@acco...	Cảnh báo bảo mật	[image: Google] Mật khẩu ứng dụng đã đ...	09/06/2024 5:36...

On the right side of the window, there are four buttons: Login, Refresh, Send Mail, and Logout.

GIAO DIỆN SENDMAIL.CS

The screenshot shows the 'Send Mail' window. It has a blue title bar and a light gray background. The window contains the following fields and controls:

- From:** phtkhang1502@gmail.com
- Name:** Khang
- To:** 22520624@gm.uit.edu.vn
- Subject:** ĐÂY LÀ MAIL TEST
- HTML:** ☐ HTML
- Body:** xin chào
- Attachment:** C:\Users\ACER\OneDrive\Computer\UI
- Buttons:** Browse, Send

MAIL KHI NHẬN**GIAO DIỆN CỦA VIEWMAIL.CS**

MENUMAIL.CS

Constructor với 0 tham số đầu vào của Form MenuMail

```
1 reference
public MenuMail()
{
    InitializeComponent();
    // Thiết lập giá trị mặc định cho các trường máy chủ và cổng
    tbImapPort.Text = "993";
    tbImapServer.Text = "imap.gmail.com";
    tbSmttPort.Text = "587";
    tbSmttServer.Text = "smtp.gmail.com";
    txtPassword.Text = "tsaq jgjs sgkr snkn";
    txtEmail.Text = "phtkhang1502@gmail.com";

    LoginDisabled(); // Vô hiệu hóa các nút khi chưa đăng nhập
}
1 reference
```

Các hàm LoginEnabled() và LoginDisabled() nhằm điều chỉnh UI khi chuyển đổi trạng thái “Đăng nhập” và “Đăng xuất”

```
1 reference
private void LoginEnabled()
{
    // Kích hoạt các nút khi đăng nhập thành công
    btRefresh.Enabled = true;
    btSendMail.Enabled = true;
    btLogout.Enabled = true;
}

2 references
private void LoginDisabled()
{
    // Vô hiệu hóa các nút khi chưa đăng nhập
    btRefresh.Enabled = false;
    btSendMail.Enabled = false;
    btLogout.Enabled = false;
}
1 reference
```


Hàm kích hoạt khi button “Login” được click: dùng để đăng nhập bằng mail chỉ định và tải lên cái mail vào listview thông qua hàm LoadEmails()

```
1 reference
private void btnLogin_Click(object sender, EventArgs e)
{
    email = txtEmail.Text;
    password = txtPassword.Text;

    try
    {
        // Khởi tạo ImapClient và kết nối tới máy chủ IMAP
        imapClient = new ImapClient();
        imapClient.Connect(tbImapServer.Text, Convert.ToInt32(tbImapPort.Text), true);
        imapClient.Authenticate(email, password);
        LoadEmails(); // Tải email

        LoginEnabled(); // Kích hoạt các nút sau khi đăng nhập thành công

        MessageBox.Show("Đăng nhập thành công", "Thành công", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Đăng nhập gặp trục trặc: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Hàm LoadMails(): dùng để tải lên dữ liệu của các mail đã triết xuất từ imapclient lên listview, ngoài ra thiết lập thêm thanh progressbar khi tải mail

```
2 references
private void LoadEmails()
{
    try
    {
        var inbox = imapClient.Inbox;
        inbox.Open(FolderAccess.ReadOnly);

        listViewEmails.Items.Clear();

        for (int i = 0; i < inbox.Count; i++)
        {
            var message = inbox.GetMessage(i);
            var item = new ListViewItem(new[]
            {
                message.From.ToString(),
                message.Subject,
                GetMessageContent(message),
                message.Date.ToString()
            });
            listViewEmails.Items.Add(item);
            progressBar1.Maximum = inbox.Count - 1;
            Invoke(new Action(() => progressBar1.Value = i));
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Failed to load emails: {ex.Message}", "Lỗi", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Phương thức kích hoạt btnRefresh_Click khi có sự kiện button “Refresh” được click: dùng để tải lại listview

Phương thức kích hoạt btnSendMail_Click khi có sự kiện button “Send Mail” được click: dùng để mở form SendMail.cs

```
1 reference
private void btnRefresh_Click(object sender, EventArgs e)
{
    LoadEmails(); // Tải lại email khi nhấn nút Refresh
}

private void btnSendMail_Click(object sender, EventArgs e)
{
    var form2 = new SendMail(email, password, tbSmtServer.Text, Convert.ToInt32(tbSmtPort.Text));
    form2.Show();
}
```

Phương thức kích hoạt khi có sự kiện nhấn đúp chuột trái vào 1 Mail trong listview: lấy thông tin mail đó làm tham số cho Constructor ViewMail để tạo Form ViewMail.cs

```
1 reference
private void listViewEmails_DoubleClick(object sender, EventArgs e)
{
    if (listViewEmails.SelectedItems.Count > 0)
    {
        var index = listViewEmails.SelectedItems[0].Index;
        var message = imapClient.Inbox.GetMessage(index);

        var form3 = new ViewMail(email, password, tbSmtServer.Text, Convert.ToInt32(tbSmtPort.Text), message);
        form3.Show();
    }
}
```

Hàm GetMessageContent(MimeMessage) dùng để lấy thông tin body của Mail nếu nó thuộc dạng text hay html

```
1 reference
private string GetMessageContent(MimeMessage message)
{
    var textBody = message.TextBody ?? "";
    var htmlBody = message.HtmlBody ?? "";

    return !string.IsNullOrEmpty(textBody) ? textBody : htmlBody;
}
```

- Phương thức kích hoạt khi có sự kiện button “Log out” được click: dùng để đăng xuất khỏi mail
 - Ngắt kết nối IMAP + SMTP
 - Reset UI

```
1 reference
private void btLogout_Click(object sender, EventArgs e)
{
    try
    {
        if (imapClient?.IsConnected == true)
        {
            imapClient.Disconnect(true); // Ngắt kết nối ImapClient
            imapClient.Dispose();
        }

        if (smtpClient?.IsConnected == true)
        {
            smtpClient.Disconnect(true); // Ngắt kết nối SmtClient
            smtpClient.Dispose();
        }

        email = string.Empty;
        password = string.Empty;
        listViewEmails.Items.Clear();
        progressBar1.Value = 0;

        LoginDisabled(); // Vô hiệu hóa các nút sau khi đăng xuất

        MessageBox.Show("Đăng xuất thành công", "Thành công", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Đăng xuất gặp trục trặc: {ex.Message}", "Lỗi", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

SENDMAIL.CS

Khai báo constructor SendMail với 4 đối số

```
1 reference
public SendMail(string email, string password, string smtpServer, int smtpPort)
{
    InitializeComponent();
    this.email = email;
    this.password = password;
    this.smtpServer = smtpServer;
    this.smtpPort = 465; // sử dụng port 465 thay vì 587
    // bị lỗi MailKit.Security.SslHandshakeException:
    // 'An error occurred while attempting to establish an SSL or TLS connection
    tbFrom.Text = email;
}
```

- Phương thức kích hoạt khi có sự kiện button “Send” được click”: gửi mail cho địa chỉ mail chỉ định, thông tin gửi bao gồm
 - Tên và địa chỉ của người gửi
 - Địa chỉ của người nhận
 - Nội dung của mail (có thể gửi ở 2 dạng HTML và text)
 - Tập đính kèm

```

1 reference
private void btnSend_Click(object sender, EventArgs e)
{
    var message = new MimeMessage();
    message.From.Add(new MailboxAddress(tbName.Text, tbFrom.Text));
    message.To.Add(new MailboxAddress("", tbTo.Text));
    message.Subject = tbSubject.Text;

    var bodyBuilder = new BodyBuilder();
    if (chkHtml.Checked) // phân biệt Body HTML
    {
        bodyBuilder.HtmlBody = tbBody.Text;
        smtpPort = 465;
    }
    else
    {
        bodyBuilder.TextBody = tbBody.Text;
    }

    if (!string.IsNullOrEmpty(tbAttachment.Text))
    {
        bodyBuilder.Attachments.Add(tbAttachment.Text);
    }

    message.Body = bodyBuilder.ToMessageBody();

    using (var client = new SmtplibClient())
    {
        client.Connect(smtpServer, 465, true);
        client.Authenticate(email, password);
        client.Send(message);
        client.Disconnect(true);
    }

    MessageBox.Show("Email gửi thành công !");
}

```

Phương thức kích hoạt khi có sự kiện button “Browse” được click: dùng để mở dialog và chọn đường dẫn file đính kèm

```

1 reference
private void btnBrowse_Click(object sender, EventArgs e)
{
    using (var dialog = new OpenFileDialog())
    {
        if (dialog.ShowDialog() == DialogResult.OK)
        {
            tbAttachment.Text = dialog.FileName;
        }
    }
}

```

VIEWMAIL.CS

Thiết lập constructor ViewMail với 5 đối số đầu vào

```
public ViewMail(string email, string password, string smtpServer, int smtpPort, MimeMessage message)
{
    InitializeComponent();

    Email = email;
    Pass = password;
    Smtpserver = smtpServer;
    Smtpport = smtpPort;

    tbFrom.Text = message.From.ToString();
    tbTo.Text = message.To.ToString();
    tbSubject.Text = message.Subject;
    tbBody.Text = message.TextBody;
}
```

Phương thức kích hoạt khi button “Reply” được click: dùng để trả lời lại Mail đang được mở trong cửa ViewMail hiện tại (mở form SendMail.cs)

```
1 reference
private void btReply_Click(object sender, EventArgs e) // Mở form SendMail.cs
{
    string input = tbFrom.Text;
    string extractedmail = ExtractEmail(input);
    var form = new SendMail(Email, Pass, extractedmail, Smtpserver, tbSubject.Text);
    form.Show();
}
```

Hàm ExtractMail(string) dùng để trích xuất mail từ chuỗi nhập vào (ví dụ: Phạm Huỳnh Tấn, Khang <22520624@gm.uit.edu.vn>, thì kết quả trả về là 22520624@gm.uit.edu.vn)

```
1 reference
private string ExtractEmail(string input) // trích xuất mail từ format ( Tên <Mail>)
{
    string pattern = @"<([^\>]+)>";
    Match match = Regex.Match(input, pattern);

    if (match.Success)
    {
        return match.Groups[1].Value;
    }

    return null;
}
```

YÊU CẦU CHUNG

1) Đánh giá

- Chuẩn bị tốt các yêu cầu đặt ra trong bài thực hành.
- Sinh viên hiểu và tự thực hiện được bài thực hành, trả lời đầy đủ các yêu cầu đặt ra.
- Nộp báo cáo kết quả chi tiết những đã thực hiện, quan sát thấy và kèm ảnh chụp màn hình kết quả (*nếu có*); giải thích cho quan sát (*nếu có*).
- Sinh viên báo cáo kết quả thực hiện và nộp bài.

2) Báo cáo

- File **.PDF** hoặc **.docx**. Tập trung vào nội dung, giải thích.
- Nội dung trình bày bằng Font chữ **Times New Romans/** hoặc font chữ của mẫu báo cáo này (UTM Avo)– cỡ chữ 13. Canh đều (Justify) cho văn bản. Canh giữa (Center) cho ảnh chụp.
- Đặt tên theo định dạng: LabX_MSSV1_MSSV2. (trong đó X là Thứ tự buổi Thực hành).

Ví dụ: Lab01_21520001_21520002

- Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

Bài sao chép, trễ, ... sẽ được xử lý tùy mức độ vi phạm.

HẾT