

# BÁO CÁO THỰC HÀNH

Môn học: Lập trình mạng căn bản

Buổi báo cáo: Lab 04

Tên chủ đề: Working with Web Server in C#

GVHD: Nguyễn Xuân Hà

Ngày thực hiện: 21/05/2024

## 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	6 ngày
Link Video thực hiện (nếu có)	<a href="https://github.com/VitalsZen">https://github.com/VitalsZen</a>
Ý kiến (nếu có) + Khó khăn + Đề xuất ...	
Điểm tự đánh giá	10

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

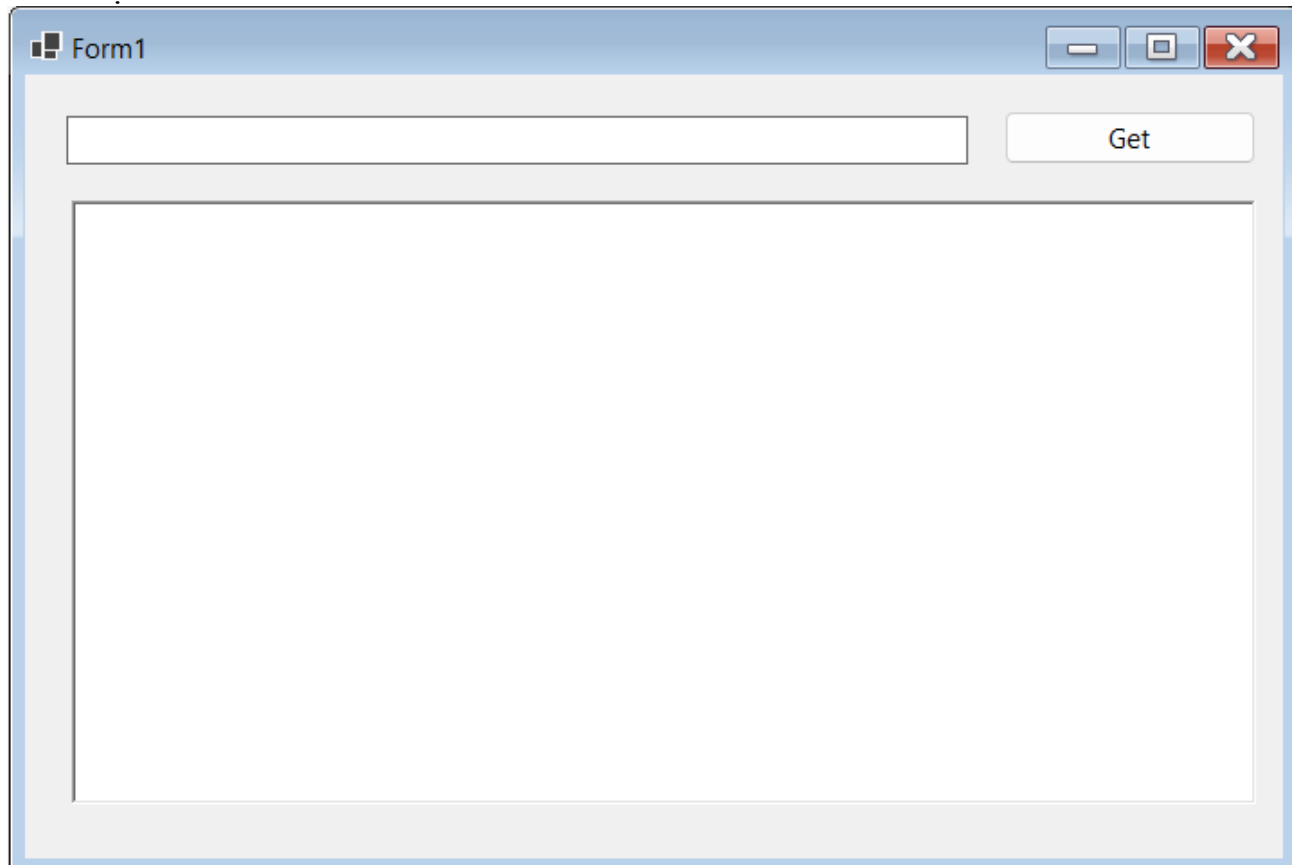
## BÁO CÁO CHI TIẾT

## Contents

Bài 01: Viết chương trình hiển thị nội dung HTML của một trang web bất kỳ:.....	3
Bài 2: Viết chương trình download nội dung trang web bất kỳ từ một địa chỉ URL bất kỳ và ghi thành file HTML, sau đó hiển thị nội dung trang web lên form. ....	4
Bài 3: Viết chương trình hoạt động như một Web Browser cơ bản.....	5
Bài 4: Quản lý phòng vé (phiên bản số 4).....	8
Bài 5: HTTP POST - Viết chương trình cho phép đăng nhập vào ứng dụng Web thông qua API được cung cấp sẵn.....	14
Bài 6: HTTP GET - Viết chương trình hiển thị thông tin người dùng hiện tại đang đăng nhập vào ứng dụng Web thông qua API được cung cấp sẵn. ....	15
Bài 7: Hôm nay ăn gì? (phiên bản số 4).....	18
YÊU CẦU CHUNG.....	30

Bài 01: Viết chương trình hiển thị nội dung HTML của một trang web bất kỳ:

Giao diện



**private void btGet\_Click(object sender, EventArgs e):** Hàm kích hoạt khi click của button Get

**private string getHTML(string szUrl):** Hàm lấy và đọc nội dung file html với tham số là szUrl

```

12 private void btGet_Click(object sender, EventArgs e)
13 {
14     //TestGetAllEndPointWithUrl(tbGetUrl.Text);
15     rtbGetContent.Text = getHTML(tbGetUrl.Text);
16 }
17
18 1 reference
19 private string getHTML(string szUrl) //lấy nội dung html
20 {
21     WebRequest request = WebRequest.Create(szUrl);
22     WebResponse response = request.GetResponse();
23
24     Stream dataStream = response.GetResponseStream();
25     StreamReader sr = new StreamReader(dataStream);
26     string responseFromServer = sr.ReadToEnd();
27
28     response.Close();
29     return responseFromServer;
30 }

```

Bài 2: Viết chương trình download nội dung trang web bất kỳ từ một địa chỉ URL bất kỳ và ghi thành file HTML, sau đó hiển thị nội dung trang web lên form.

Giao diện

**private void btDownload\_Click(object sender, EventArgs e):** Hàm kích hoạt khi được click của button Download

**private string getHTML(string szUrl):** Hàm lấy và đọc nội dung file html với tham số là szUrl

```
1 reference
private void btDownload_Click(object sender, EventArgs e)
{
    rtbGetContent.Text = getHTML(tbGetUrl.Text);
    getHTMLFile(tbGetUrl.Text, tbDownloadUrl.Text);
}

1 reference
private string getHTML(string szUrl) //lấy và đọc nội dung file HTML bằng lớp WebRequest
{
    WebRequest request = WebRequest.Create(szUrl);
    WebResponse response = request.GetResponse();

    Stream dataStream = response.GetResponseStream();
    StreamReader sr = new StreamReader(dataStream);
    string responseFromServer = sr.ReadToEnd();

    response.Close();
    return responseFromServer;
}
```

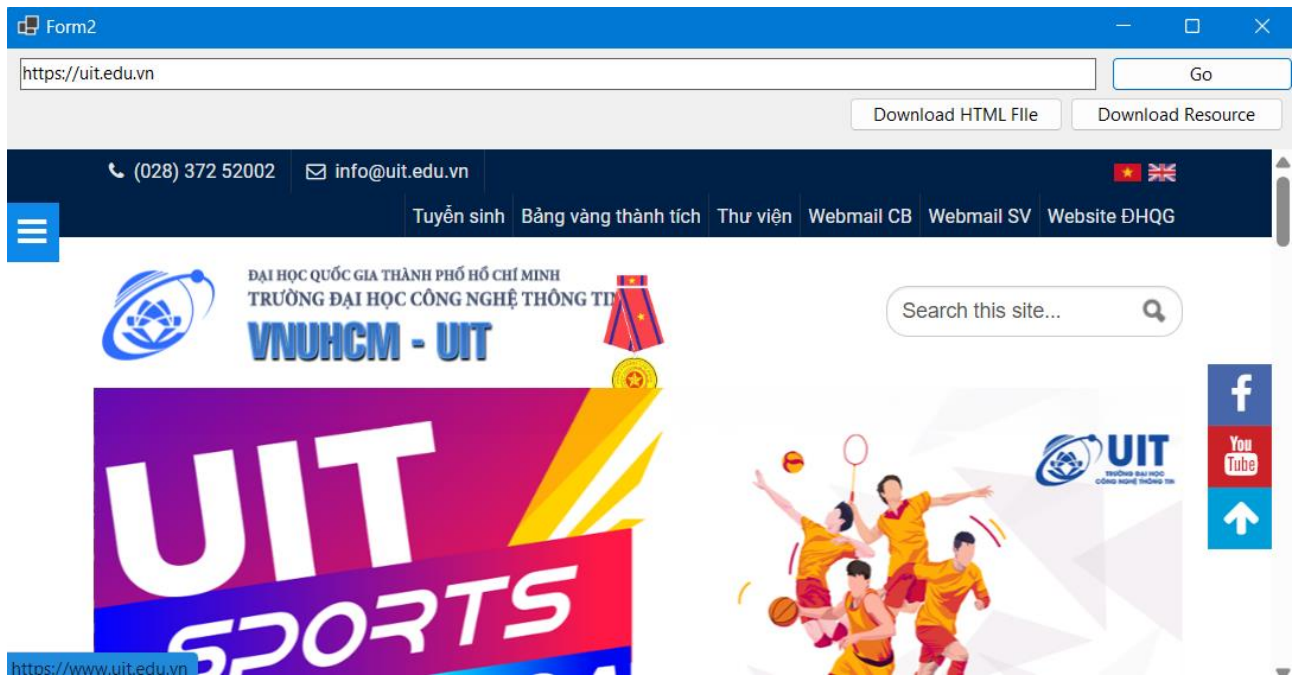
**private void getHTMLFile(string Url, string FileUrl):** Hàm lưu nội dung file HTML với tham số là Url là đường dẫn đến web chỉ định và FileUrl là đường dẫn của file lưu về

```
1 reference
private void getHTMLFile(string Url, string FileUrl) //lưu nội dung file HTML bằng lớp WebClient
{
    WebClient myclient = new WebClient();
    Stream response = myclient.OpenRead(Url);
    myclient.DownloadFile(Url, FileUrl);
}
```

### Bài 3: Viết chương trình hoạt động như một Web Browser cơ bản

Có các tính năng sau:

- Xem nội dung Website
- Download File html
- Download Resource của Website



**private void Form\_Resize(object sender, EventArgs e):** Hàm thay đổi kích thước giao diện của các controls khi form thay đổi kích thước

**private void btConnect\_Click(object sender, EventArgs e):** Hàm kích hoạt khi click của button Go -> truy cập web với url có sẵn ở textbox tbConnectUrl

```
1 reference
public Form1()
{
    InitializeComponent();
    this.Resize += new System.EventHandler(this.Form_Resize); // thay đổi kích thước Form

    //webView21.NavigationStarting += EnsureHttps; // các lệnh điều hướng webview để truy cập các url an toàn
    //InitializeAsync();
}

// async void InitializeAsync() // hàm báo cảnh báo khi truy cập url không an toàn ...
1 reference
private void Form_Resize(object sender, EventArgs e) // Đổi kích thước giao diện các controls khi kích thước form thay đổi
{
    webView21.Size = this.ClientSize - new System.Drawing.Size(webView21.Location);
    btConnect.Left = this.ClientSize.Width - btConnect.Width;
    tbConnectUrl.Width = btConnect.Left - tbConnectUrl.Left;
}

1 reference
private void btConnect_Click(object sender, EventArgs e)
{
    if (webView21 != null && webView21.CoreWebView2 != null)
    {
        webView21.CoreWebView2.Navigate(tbConnectUrl.Text);
    }
}
```

**private void btDownloadResource\_Click(object sender, EventArgs e):** Hàm kích hoạt khi click của button Download Resource → điều hướng để kích hoạt hàm DownloadAllImages(string)

```

1 reference
private void btDownloadResource_Click(object sender, EventArgs e) // Hàm event click dùng để tải
                                                                    // các hình ảnh đang sẵn có trên web hiện thời
{
    string url = tbConnectUrl.Text;
    if (!string.IsNullOrEmpty(url))
    {
        DownloadAllImages(url);
        MessageBox.Show("Images downloaded from " + url);
    }
}

```

**private void DownloadAllImages(string url):** Hàm xác định các node hình ảnh của nội dung HTML của trang web sau đó kiểm tra xác thực và url cần thiết để tiến hành thực thi hàm DownloadImage(string) để tải hình ảnh

```

1 reference
private void DownloadAllImages(string url) // hàm dùng để lưu tất cả các node //img của file html của web hiện thời
{
    var html = new HtmlWeb();
    var document = html.Load(url);

    var imageNodes = document.DocumentNode.SelectNodes("//img");
    if (imageNodes != null)
    {
        foreach (var img in imageNodes)
        {
            string src = img.GetAttributeValue("src", null);
            if (!string.IsNullOrEmpty(src))
            {
                if (!Uri.IsWellFormedUriString(src, UriKind.Absolute))
                {
                    Uri baseUri = new Uri(url);
                    Uri absoluteUri = new Uri(baseUri, src);
                    src = absoluteUri.ToString();
                }
                DownloadImage(src);
            }
        }
    }
}

```

**private void DownloadImage(string url):** Hàm dùng để lưu ảnh về máy với tham số là đường dẫn URL

```

1 reference
private void DownloadImage(string url) // lưu ảnh về máy
{
    WebClient client = new WebClient();
    Uri uri = new Uri(url);
    string fileName = Path.GetFileName(uri.LocalPath);
    client.DownloadFile(uri, fileName);
}

```

**private void btDownloadHTMLContent\_Click(object sender, EventArgs e):** Hàm kích hoạt khi click của button Download HTML File → tạo đường dẫn lưu file + điều hướng thực hiện hàm getHTMLFile(string,string)

**private void getHTMLFile(string Url, string FileUrl):** Hàm dùng để lưu file nội dung HTML của web chỉ định với tham số string Url là đường dẫn của web đó

```

1 reference
private void btDownloadHTMLContent_Click(object sender, EventArgs e) // Hàm event click dùng để tải các file HTML
{
    // tạo tên file json để lưu về máy ( ví dụ: https://uit.edu.vn -> uit.edu.vn.json )
    string SaveUrl = tbConnectUrl.Text.Replace("https://", "").Replace("/", ".") + ".html";
    getHTMLFile(tbConnectUrl.Text, SaveUrl); // gọi hàm
    MessageBox.Show("HTML file downloaded as " + SaveUrl);
}

1 reference
private void getHTMLFile(string Url, string FileUrl) // Hàm dùng để lưu file html của url chỉ định vào
// về đường dẫn trên máy tính Fileurl
{
    WebClient myclient = new WebClient();
    Stream response = myclient.OpenRead(Url);
    myclient.DownloadFile(Url, FileUrl);
}

```

## Bài 4: Quản lý phòng vé (phiên bản số 4)

Lấy ý tưởng và kế thừa từ bài 5 - bài thực

hành số 2. Viết một ứng dụng cho phép draw dữ liệu từ một website đặt vé xem phim. Dữ liệu tóm tắt về thông tin của bộ phim sẽ được lưu trữ dưới dạng JSON file. Hệ thống giúp hỗ trợ nhân viên đặt vé cho khách hàng, thông tin về khách hàng, vé và tiền thanh toán sẽ được thông báo ngay khi đặt vé.

Lưu ý:

- Website gợi ý: <https://betacinemas.vn/phim.htm>
- Thanh ProgressBar để hiển thị tiến trình trích xuất thông tin từ website.
- Nếu chọn vào banner của 1 bộ phim bất kỳ, truy cập đường dẫn đến bộ phim đó tại website gợi ý để hiển thị thông tin chi tiết của bộ phim.

### Giao diện của MainForm



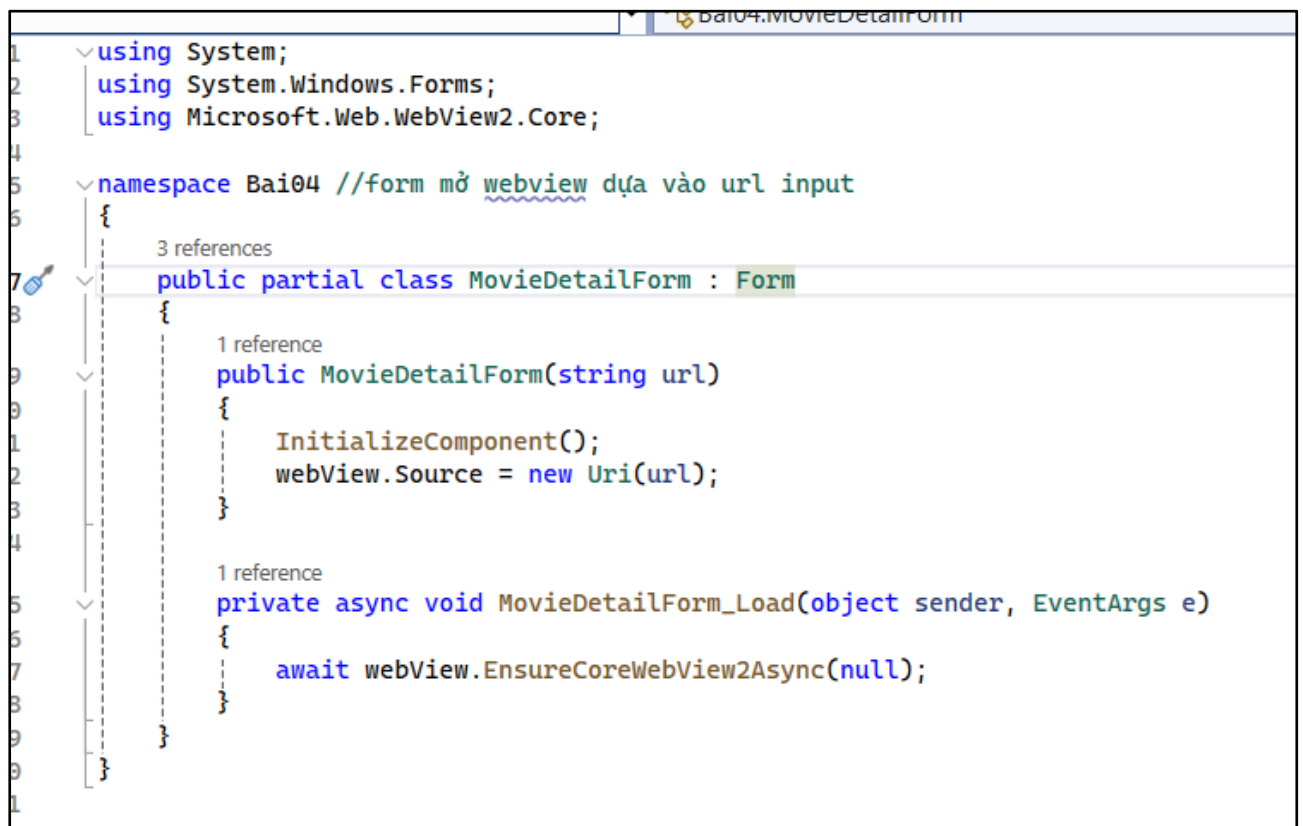


Giao diện của Webview sau khi click vào panel của 1 phim bất kì



### MovieDetailForm.cs

Form dùng để mở webview với constructor chứa 1 đối số là string url (đường dẫn đến web chỉ định)



MainForm.cs

**private void Form1\_Load(object sender, EventArgs e):** Hàm định dạng các controls khi mới load MainForm

- **movieFlowPanel:** dùng để sắp đặt 1 bố cục các panel sẽ được khai triển khi triết xuất từng bộ phim từ web
- **progressBar:** thanh quá trình thể hiện tiến độ triết xuất dữ liệu từ web và quá trình load của các bộ phim lên MainForm

```
{
    private FlowLayoutPanel movieFlowPanel;
    1 reference
    public Form1()
    {
        InitializeComponent();
    }

    1 reference
    private void Form1_Load(object sender, EventArgs e) // Hàm định dạng các controls mới khi load Form
    {
        movieFlowPanel = new FlowLayoutPanel
        {
            Dock = DockStyle.Fill,
            AutoScroll = true,
            FlowDirection = FlowDirection.TopDown, //Đẩy các panel theo chiều hướng xuống
            WrapContents = false
        };
        Controls.Add(movieFlowPanel);

        progressBar = new ProgressBar
        {
            Dock = DockStyle.Top,
            Maximum = 100,
            Value = 0
        };
        Controls.Add(progressBar);

        LoadMoviesAsync(); // Gọi hàm
    }
}
```

**private async Task LoadMoviesAsync():** Hàm điều hướng các hàm con trong quá trình load phim

- **FetchMoviesAsync():** triết xuất dữ liệu phim
- **SaveMoviesToJson(movies):** lưu dữ liệu phim thành file json về máy
- **DisplayMovies(movies):** Load phim lên MainForm từ dữ liệu file json

```
1 reference
private async Task LoadMoviesAsync() // Hàm điều hướng các hàm con trong quá trình load phim
{
    try
    {
        var movies = await FetchMoviesAsync(); // triết xuất dữ liệu phim
        SaveMoviesToJson(movies); // lưu dữ liệu phim thành file json về máy
        DisplayMovies(movies); // Load phim lên MainForm từ dữ liệu file json
    }
    catch (Exception ex)
    {
        MessageBox.Show($"An error occurred: {ex.Message}");
    }
}
```

**private async Task<List<Movie>> FetchMoviesAsync()** :Hàm draw dữ liệu từ web sử dụng HTMLAgilityPack đã cài đặt trong Nuget của project

- Xác định URL mục tiêu
- Lấy nội dung file html và load bằng HTMLAgilityPack
  - Xác định các Nodes phim
  - Với mỗi Node phim, xác định các thành phần con (tên phim, link truy cập, link hình ảnh, thời lượng, thể loại) gán nội dung trên vào đối tượng thuộc lớp movie đã cài đặt sẵn
- Hàm trả về 1 danh sách gồm các đối tượng thuộc lớp movie

```
private async Task<List<Movie>> FetchMoviesAsync() // triết xuất phim
{
    var movies = new List<Movie>();
    string url = "https://betacinemas.vn/phim.htm";
    HttpClient client = new HttpClient();
    var response = await client.GetStringAsync(url); // request GET cho url

    var htmlDoc = new HtmlAgilityPack.HtmlDocument();
    htmlDoc.LoadHtml(response); // Load file html

    var MoviesNode = htmlDoc.DocumentNode.SelectNodes("//div[contains(@class, 'col-lg-4 col-md-4 col-sm-8 col-xs-16 pad' +
        "ding-right-30 padding-left-30 padding-bottom-30')]");

    //lấy node của movie
    if (MoviesNode == null || MoviesNode.Count == 0)
    {
        throw new Exception("No movies found. The XPath query did not return any results.");
    }

    progressBar.Maximum = MoviesNode.Count;
    progressBar.Value = 0;

    foreach (var node in MoviesNode) // triết xuất node con từ node của MoviesNode
    {
        var titleNode = node.SelectSingleNode("./h3/a"); // xác định node "Tiêu đề phim"
        var linkNode = node.SelectSingleNode("./h3/a"); // xác định node "link truy cập"
        var imgNode = node.SelectSingleNode("./img[@class='img-responsive border-radius-20']"); //xác định node "hình ảnh"

        if (titleNode != null && linkNode != null && imgNode != null)
        {
            var detailUrl = "https://betacinemas.vn" + linkNode.Attributes["href"].Value;
            var detailResponse = await client.GetStringAsync(detailUrl);
            var detailDoc = new HtmlAgilityPack.HtmlDocument();
            detailDoc.LoadHtml(detailResponse);

            var genreNode = detailDoc.DocumentNode.SelectSingleNode("//ul[@class='list-unstyled font-lg font-family-sans font-sm-15 font-xs-14']/li[1]");
            var durationNode = detailDoc.DocumentNode.SelectSingleNode("//ul[@class='list-unstyled font-lg font-family-sans font-sm-15 font-xs-14']/li[2]");
            // xác định các node thể loại và thời lượng xem nhưng không load lên được
            //var genre = genreNode?.InnerText.Replace("Thể loại:", "").Trim();
            //var durationText = durationNode?.InnerText.Replace("Thời lượng:", "").Replace("phút", "").Trim();
            //int.TryParse(durationText, out int duration);

            var movie = new Movie
            {
                Title = titleNode.InnerText.Trim(),
                DetailUrl = detailUrl,
                ImageUrl = imgNode.Attributes["src"].Value,
                //Duration = duration,
                //Genre = genre ?? "N/A"
            };
            movies.Add(movie);
        }
        progressBar.Value++; // ++ progressBar với mỗi node của 1 phim load hoàn thành
    }
    return movies;
}
```

**private void SaveMoviesToJson(List<Movie> movies):** Hàm dùng để chuyển thông tin của danh sách các đối tượng thuộc lớp movie vào file json và lưu về máy

```
private void SaveMoviesToJson(List<Movie> movies) //lưu thông tin phim vào file
{
    string json = JsonConvert.SerializeObject(movies, Formatting.Indented);
    System.IO.File.WriteAllText("movies.json", json);
}
```

**private void DisplayMovies(List<Movie> movies):** Hàm dùng để biểu diễn thông tin của danh sách các đối tượng thuộc lớp phim thu hồi dữ liệu từ web thành các hình ảnh và thông tin cụ thể

( mỗi phim sẽ tương ứng với 1 panel gồm: 1 pictureBox( hình ảnh của phim), 1 LabelCha( tiêu đề phim ), 1 LabelCon ( link truy cập vào bộ phim đó) ) và các thông tin trên đều có thể tương tác thông qua event click khi nhấn vào bất kì controls liệt kê phía trên của từng bộ phim sẽ dẫn đến trang web riêng của bộ phim đó)

```

1 reference
private void DisplayMovies(List<Movie> movies) // show các phim
{
    foreach (var movie in movies)
    { // định dạng các controls trong panel
        var pictureBox = new PictureBox
        {
            ImageLocation = movie.ImageUrl,
            SizeMode = PictureBoxSizeMode.StretchImage,
            Width = 150,
            Height = 200,
            Dock = DockStyle.Left,
            Cursor = Cursors.Hand
        };

        var LabelCha = new Label
        {
            Text = movie.Title.Trim(),
            Font = new Font("Arial", 19F, FontStyle.Bold, GraphicsUnit.Point, 0),
            ForeColor = Color.FromArgb(0, 192, 0),
            AutoSize = true,
            TextAlign = ContentAlignment.MiddleLeft,
            Dock = DockStyle.Top,
            Padding = new Padding(10),
            Cursor = Cursors.Hand
        };

        var LabelCon = new Label
        {
            Text = $"{movie.DetailUrl}",
            Font = new Font("Arial", 13F, FontStyle.Italic, GraphicsUnit.Point, 0),
            AutoSize = true,
            TextAlign = ContentAlignment.MiddleLeft,
            Dock = DockStyle.Fill,
            Padding = new Padding(10),
            Cursor = Cursors.Hand
        };
    }
}

```

```

var panel = new Panel
{
    Width = movieFlowPanel.ClientSize.Width - 20,
    Height = 250,
    Margin = new Padding(5),
    BorderStyle = BorderStyle.FixedSingle
};

panel.Controls.Add(Labelcon);
panel.Controls.Add(LabelCha);
panel.Controls.Add(pictureBox);

// các event của label + pictureBox
Labelcon.MouseEnter += (sender, e) => // rê chuột vào là gạch dưới với in nghiêng
{
    Labelcon.Font = new Font(Labelcon.Font, FontStyle.Italic | FontStyle.Underline);
};
Labelcon.MouseLeave += (sender, e) => // quay về bình thường
{
    Labelcon.Font = new Font(Labelcon.Font, FontStyle.Italic);
};
Labelcon.Click += (sender, e) =>
{
    ShowMovieDetail(movie.DetailUrl);
};
// tương tự Label con
LabelCha.MouseEnter += (sender, e) =>
{
    LabelCha.Font = new Font(LabelCha.Font, FontStyle.Bold | FontStyle.Underline);
};
LabelCha.MouseLeave += (sender, e) =>
{
    LabelCha.Font = new Font(LabelCha.Font, FontStyle.Bold);
};

LabelCha.Click += (sender, e) =>
{
    ShowMovieDetail(movie.DetailUrl);
};

pictureBox.Click += (sender, e) =>
{
    ShowMovieDetail(movie.DetailUrl);
};

movieFlowPanel.Controls.Add(panel);
}

```

**private void ShowMovieDetail(string detailUrl):** Hàm sử dụng để khởi tạo Form trang web với tham số là URL của trang web đó

```

3 references
private void ShowMovieDetail(string detailUrl)
{
    var movieDetailForm = new MovieDetailForm(detailUrl);
    movieDetailForm.Show();
}

```

Bài 5: HTTP POST - Viết chương trình cho phép đăng nhập vào ứng dụng Web thông qua API được cung cấp sẵn.

Giao diện

Form1

URL:

Username:

Password:

Connect

Bearer  
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImBoYXRwdCIsImV4cCI6MTcxNjQ0NTA3M30.h6inunf9m3-bl7ioQOJ8Zec6Kyuw-5wkGFWP-vluV84  
Đăng nhập thành công

**private async Task ConnectToApiAsync():** Hàm dùng để kết nối đến API với yêu cầu HTTP POST để lấy token phê duyệt

- Xác định request body ( username, password)
- Xác định API
- Tạo liên kết bằng lớp HttpClient và thực hiện gửi yêu cầu POST cho API
- Tạo 1 biến kiểu string để nhận gói thông tin API gửi về (token)
- Tách riêng AccessToken và TokenType bằng phần mở rộng Newtonsoft.Json được cài đặt bằng Nuget (JsonObject)
- Đẩy 2 thông tin của token lên RichTextBox



```

1 reference
private async Task ConnectToApiAsync()// hàm dùng để kết nối đến API
{
    string url = tbUrl.Text;
    string username = tbUsername.Text;
    string password = tbPassword.Text;

    var data = new MultipartFormDataContent(); // chuẩn bị data để kèm theo request
    data.Add(new StringContent(username), "username");
    data.Add(new StringContent(password), "password");

    try
    {
        using (HttpClient client = new HttpClient())
        {
            HttpResponseMessage response = await client.PostAsync(url, data); // request HTTP POST đến API
            string responseContent = await response.Content.ReadAsStringAsync();

            if (response.IsSuccessStatusCode)
            {
                var jsonResponse = JObject.Parse(responseContent); // nhận token từ API
                string tokenType = jsonResponse["token_type"].ToString();
                string accessToken = jsonResponse["access_token"].ToString();

                rtbContent.Text = $"Bearer {accessToken}";

                rtbContent.Text += '\n' + "Đăng nhập thành công";
                MessageBox.Show("Đăng nhập thành công", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            else
            {
                var jsonResponse = JObject.Parse(responseContent);
                string detail = jsonResponse["detail"].ToString();
                MessageBox.Show(detail, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Có lỗi xảy ra: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

```

Bài 6: HTTP GET - Viết chương trình hiển thị thông tin người dùng hiện tại đang đăng nhập vào ứng dụng Web thông qua API được cung cấp sẵn.

Giao diện

The screenshot shows a Windows application window titled "Form1". It has a standard Windows title bar with minimize, maximize, and close buttons. The main area contains three input fields: "URL:" with the value "https://nt106.uit.edu.vn/auth/token", "Username:" with the value "phatpt", and "Password:" with masked characters "\*\*\*\*\*". To the right of these fields are two buttons: "Connect" and "Get User Info". Below the input fields is a text area labeled "User Info:" containing the following details: "Id: 2", "Username: phatpt", "Email: phatpt@uit.edu.vn", "Email Verified: False", "First Name: Phat", "Last Name: Phan Trung", "Avatar:", "Sex: 1", "Birthday: 2024-04-19", and "Language: english".

**private async Task ConnectToApiAsync():** Hàm tương tự Bài 5 → Dùng để lấy thông tin Token

```
1 reference
private async Task ConnectToApiAsync()
{
    // Lưu thông tin url, username, password
    string url = tbUrl.Text;
    string username = tbUsername.Text;
    string password = tbPassword.Text;

    // tạo Data lưu thông tin trên
    var Data = new MultipartFormDataContent();
    Data.Add(new StringContent(username), "username");
    Data.Add(new StringContent(password), "password");

    try
    {
        using (HttpClient client = new HttpClient())
        {
            HttpResponseMessage response = await client.PostAsync(url, Data); // gửi request cho API
            string responseContent = await response.Content.ReadAsStringAsync(); // Đọc nội dung API trả về

            if (response.IsSuccessStatusCode)
            {
                // Chuyển đổi file nhận được thành file json ( Newtonsoft.Json)
                var jsonResponse = JObject.Parse(responseContent);
                tokenType = jsonResponse["token_type"].ToString();
                accessToken = jsonResponse["access_token"].ToString();

                rtbContent.Text = $"{tokenType} {accessToken}";
                MessageBox.Show("Đăng nhập thành công", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            else
            {
                var jsonResponse = JObject.Parse(responseContent);
                string detail = jsonResponse["detail"].ToString();
                MessageBox.Show(detail, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error); // Show error message
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Có lỗi xảy ra: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

**private async void btGetUserInfo\_Click(object sender, EventArgs e):** Hàm kích hoạt khi được click của button Get User Info → điều hướng và thực thi hàm **GetUserInfoAsync()**

```
1 reference
private async void btGetUserInfo_Click(object sender, EventArgs e)
{
    await GetUserInfoAsync();
}
```



**private async Task GetUserInfoAsync():** Hàm dùng để kết nối với API nhưng dùng để lấy thông tin của người dùng với nội dung request bao gồm kèm theo token xác thực đã lấy khi đăng nhập user trước đó. Sau đó API sẽ trả về file json thông tin của người dùng

Cấu trúc API yêu cầu:

Get details of currently logged in user

AUTHORIZATIONS: ▾

OAuth2PasswordBearerWithHeader

OAuth2: OAuth2PasswordBearerWithHeader

Flow type: `password`

Token URL: `/auth/token`

Responses

✓ 200 Successful Response

RESPONSE SCHEMA: `application/json`

id	any (Id)
username	string (Username) <i>required</i>
email	string <email> (Email) <i>required</i>
email_verified	boolean (Email Verified) <i>required</i>
first_name	string (First Name)
last_name	string (Last Name)
avatar	string (Avatar)
sex	integer (Sex)
birthday	string <date> (Birthday)
language	string (Language)
phone	string (Phone)
phone_verified	boolean (Phone Verified)
is_active	boolean (Is Active)

GET `/api/v1/user/me`

`https://nt106.ultiot.vn/api/v1/user/me`

200

Content type: `application/json`

```

{
  "id": null,
  "username": "string",
  "email": "user@example.com",
  "email_verified": true,
  "first_name": "string",
  "last_name": "string",
  "avatar": "string",
  "sex": 0,
  "birthday": "2019-01-24",
  "language": "string",
  "phone": "string",
  "phone_verified": true,
  "is_active": true,
  "is_superuser": true
}

```

Copy

```
private async Task GetUserInfoAsync() // Hàm lấy thông tin User
{
    if (string.IsNullOrEmpty(accessToken)) //Kiểm tra đăng nhập chưa ( đăng nhập rồi -> Có token )
    {
        MessageBox.Show("Bạn cần đăng nhập trước", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }

    string url = "https://nt106.uitiot.vn/api/v1/user/me"; // url API mục tiêu

    try
    {
        using (HttpClient client = new HttpClient())
        {
            // Thêm header của request bằng Token để được phê duyệt cho lần request này
            client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue(tokenType, accessToken);

            HttpResponseMessage response = await client.GetAsync(url); //Gửi yêu cầu GET
            string responseContent = await response.Content.ReadAsStringAsync();

            if (response.IsSuccessStatusCode)
            {
                var jsonResponse = JObject.Parse(responseContent);
                string userInfo = $"User Info:\n" +
                    $"Id: {jsonResponse["id"]}\n" +
                    $"Username: {jsonResponse["username"]}\n" +
                    $"Email: {jsonResponse["email"]}\n" +
                    $"Email Verified: {jsonResponse["email_verified"]}\n" +
                    $"First Name: {jsonResponse["first_name"]}\n" +
                    $"Last Name: {jsonResponse["last_name"]}\n" +
                    $"Avatar: {jsonResponse["avatar"]}\n" +
                    $"Sex: {jsonResponse["sex"]}\n" +
                    $"Birthday: {jsonResponse["birthday"]}\n" +
                    $"Language: {jsonResponse["language"]}\n" +
                    $"Phone: {jsonResponse["phone"]}\n" +
                    $"Phone Verified: {jsonResponse["phone_verified"]}\n" +
                    $"Is Active: {jsonResponse["is_active"]}\n" +
                    $"Is Superuser: {jsonResponse["is_superuser"]}";
                richTextBox1.Text = userInfo; // Display user info in the RichTextBox
            }
            else
            {
                var jsonResponse = JObject.Parse(responseContent);
                string detail = jsonResponse["detail"].ToString();
                MessageBox.Show(detail, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error); // Show error message
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Có lỗi xảy ra: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error); // Show exception message
    }
}
```

## Bài 7: Hôm nay ăn gì? (phiên bản số 4).

Lấy ý tưởng và kế thừa từ bài 5 - bài thực hành số 3 và dựa trên hệ thống API đã cung cấp sẵn. Chức năng chính của ứng dụng vẫn dùng để ngẫu nhiên chọn ra 1 món ăn từ dữ liệu của bản thân hoặc dữ liệu của cả cộng đồng.

Link API: [Hôm nay ăn gì - ReDoc \(uitiot.vn\)](https://nt106.uitiot.vn)

Lưu ý:

- Các tính năng cơ bản của một ứng dụng phải được đảm bảo:

- Tạo tài khoản (1.1) đăng nhập (1.2)
- Thêm 1 món ăn mới (2)

- Xoá 1 món ăn (3)
- Chức năng hiển thị tất cả các món ăn có trên hệ thống, phân trang (4)
- Chức năng hiển thị các món ăn do bản thân tạo, phân trang. (5)
- Ngẫu nhiên trong tất cả món ăn từ cả cộng đồng. (6)
- Ngẫu nhiên trong những món ăn bản thân đóng góp (7)

Tạo 4 form để phục vụ các chức năng trên

LoginForm: (1.2)

RegisterForm: (1.1)

MainForm: (3), (4), (5), (6), (7)

AddFoodForm (2)

Giao diện của LoginForm.cs

Tạo 2 class dùng để lưu trữ thông tin log in và Token khi đăng nhập thành công để sử dụng cho các thao tác khác với API sau này

```
namespace Bai07
{
    12 references
    public static class Tokens // lưu thông tin của token kiểm duyệt
    {
        6 references
        public static string TokenType { get; set; }
        6 references
        public static string AccessToken { get; set; }
    }

    2 references
    public static class User //class dùng để lưu thông tin logged in user
    {
        2 references
        public static string LoginUser { get; set; }
    }
}
```

**private async Task ConnectToApiAsync():** Hàm dùng để kết nối tới API thông qua request POST → đăng nhập và lấy token ( giống với hàm **ConnectToApiAsync()** của bài 5)

```

1 reference
private async Task ConnectToApiAsync()
{
    string url = "https://nt106.uit.edu.vn/auth/token";
    string username = txtUsername.Text;
    string password = txtPassword.Text;

    var formData = new MultipartFormDataContent();
    formData.Add(new StringContent(username), "username");
    formData.Add(new StringContent(password), "password");

    try
    {
        using (HttpClient client = new HttpClient())
        {
            // Send a POST request with the form data
            HttpResponseMessage response = await client.PostAsync(url, formData);
            string responseContent = await response.Content.ReadAsStringAsync();

            if (response.IsSuccessStatusCode) // 200 code
            {
                User.LoginUser = username; // lưu thông tin loginuser

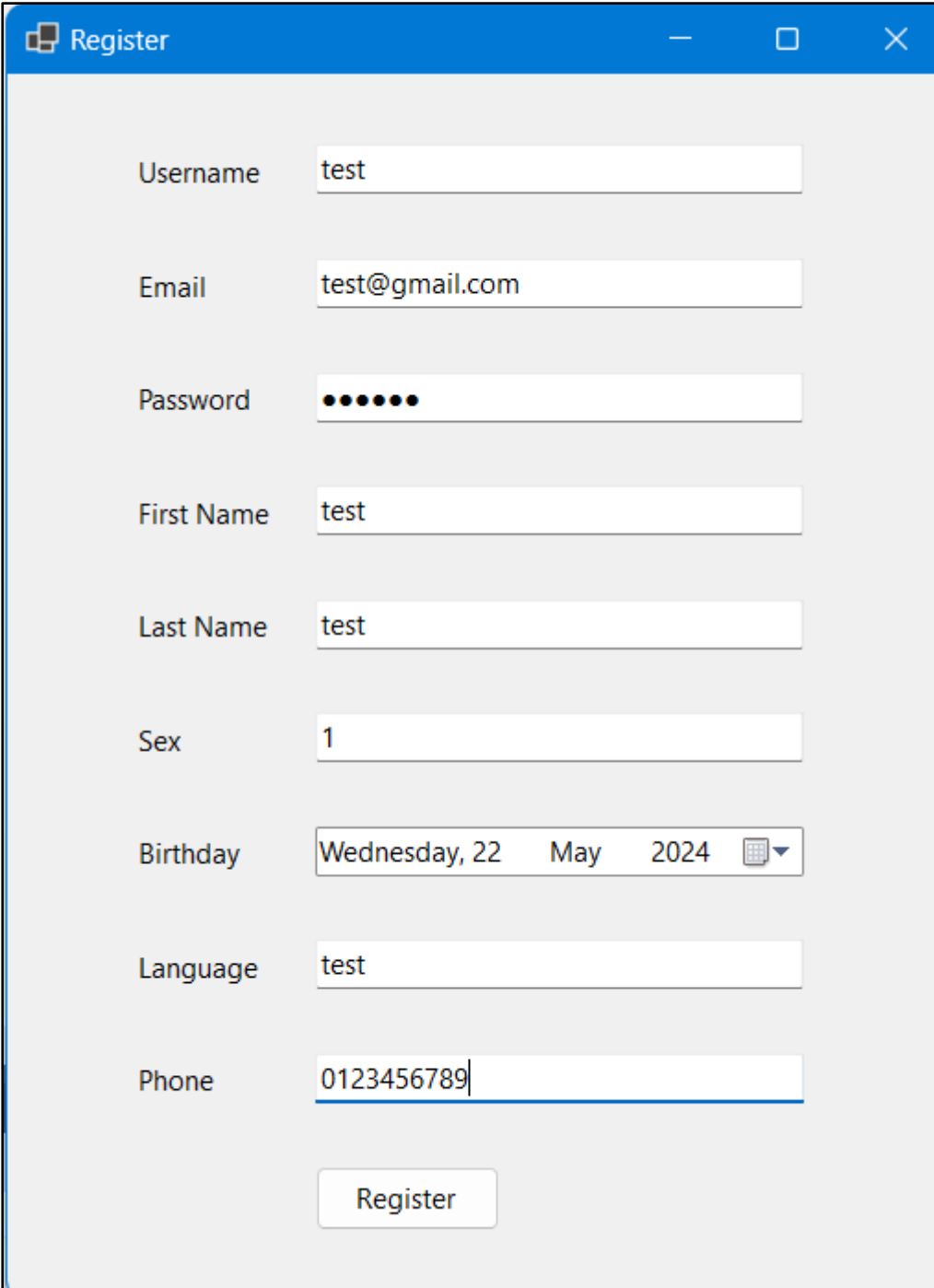
                var jsonResponse = JObject.Parse(responseContent);
                Tokens.TokenType = jsonResponse["token_type"].ToString();
                Tokens.AccessToken = jsonResponse["access_token"].ToString();

                MessageBox.Show("Đăng nhập thành công", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);

                this.Hide(); // mở MainForm
                var mainForm = new MainForm();
                mainForm.Show();
            }
            else
            {
                var jsonResponse = JObject.Parse(responseContent);
                string detail = jsonResponse["detail"].ToString();
                MessageBox.Show(detail, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Có lỗi xảy ra: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

```

Giao diện của *RegisterForm.cs*



The image shows a web browser window with a title bar that says "Register". The window contains a registration form with the following fields and values:

- Username: test
- Email: test@gmail.com
- Password: (masked with 8 dots)
- First Name: test
- Last Name: test
- Sex: 1
- Birthday: Wednesday, 22 May 2024 (with a calendar icon)
- Language: test
- Phone: 0123456789

At the bottom of the form is a button labeled "Register".

**private async void btnRegister\_Click(object sender, EventArgs e):** Hàm kích hoạt khi click của button Register

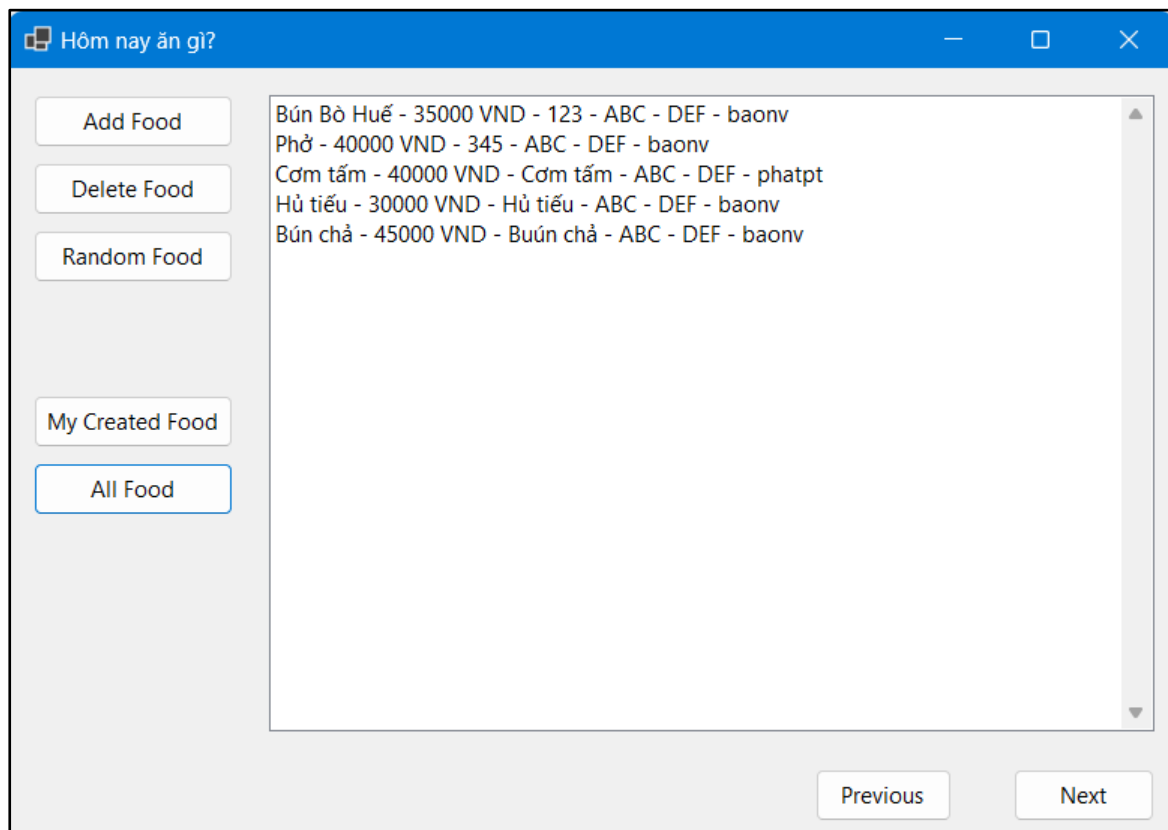
- Chuyển đổi toàn bộ thông tin nhập thành file json
- Thiết lập kết nối và gửi yêu cầu đến API bằng phương thức POST

```
1 reference
private async void btnRegister_Click(object sender, EventArgs e)
{
    var registerInfo = new // thu thập set thông tin
    {
        username = txtUsername.Text,
        email = txtEmail.Text,
        password = txtPassword.Text,
        first_name = txtFirstName.Text,
        last_name = txtLastName.Text,
        sex = int.Parse(txtSex.Text),
        birthday = dateTimePickerBirthday.Value.ToString("yyyy-MM-dd"),
        language = txtLanguage.Text,
        phone = txtPhone.Text
    };

    string json = JsonConvert.SerializeObject(registerInfo);
    var data = new StringContent(json, Encoding.UTF8, "application/json");

    using (HttpClient client = new HttpClient()) // chuyển đổi thông tin trên thành json file sau đó gửi request HTTP POST cho API
    {
        var response = await client.PostAsync("https://nt106.uit.edu.vn/api/v1/user/signup", data);
        if (response.IsSuccessStatusCode)
        {
            MessageBox.Show("Đăng ký thành công!");
            this.Close();
        }
        else
        {
            MessageBox.Show("Đăng ký thất bại!");
        }
    }
}
```

Giao diện của MainForm.cs



Thiết lập các biến dùng để thể hiện trạng thái khi thao tác trên form

```
3 references
public partial class MainForm : Form
{
    private List<Food> foodList;
    private int currentUserPage = 1;
    private int currentAllPage = 1;
    private string currentList = "AllFood"; // phân biệt tab thức ăn công đồng/ thức ăn tự tạo

    1 reference
    public MainForm()
    {
        InitializeComponent();
        this.Load += new EventHandler(MainForm_Load); // Load Main Form
    }

    1 reference
    private async void MainForm_Load(object sender, EventArgs e)
    {
        await LoadFoodDataAsync(); // gọi hàm
    }
}
```

**private async Task LoadFoodDataAsync():** Hàm dùng để tải dữ liệu món ăn của cộng đồng (cách thức hoạt động vẫn giống các hàm gọi API khác)

- Request data bao gồm
  - Header được gán Token đã có trước đó
  - Body
 

```
{ // do API yêu cầu
  "current": {number},
  "pageSize": {number}
}
```
- Gửi nội dung trên cho API với URL chỉ định
- Tạo biến chứa nội dung file json được gửi về:

```
{
  "data": [
    {
      "id": null,
      "ten_mon_an": "string",
      "gia": 0,
      "mo_ta": "string",
      "hinh_anh": "string",
      "dia_chi": "string",
      "nguoi_dong_gop": "string"
    }
  ],
  "pagination": {
    "current": 0,
    "pageSize": 0,
    "total": 0
  }
}
```

- **Lưu nội dung vào file json (do hàm này có vai trò như hàm INIT) // đây là chỗ khiến hàm này khác với hàm LoadAllFoodAsync(int currentAllPage) khi chúng đều có tác dụng là tải dữ liệu của món ăn của cả cộng đồng**

```
1 reference
private async Task LoadFoodDataAsync() // Tải dữ liệu món ăn từ API
{
    using (HttpClient client = new HttpClient())
    {
        client.DefaultRequestHeaders.Authorization = new
            System.Net.Http.Headers.AuthenticationHeaderValue(Tokens.TokenType, Tokens.AccessToken); // kèm Token vào header để được phê duyệt

        var requestBody = new
        {
            current = 1,
            pageSize = 5
        }; // nội dung request

        var json = JsonConvert.SerializeObject(requestBody);
        var content = new StringContent(json, Encoding.UTF8, "application/json");

        var response = await client.PostAsync("https://nt106.uitot.vn/api/v1/monan/all", content); // thực hiện request HTTP POST với nội dung trên

        if (response.IsSuccessStatusCode)
        {
            var jsonResponse = await response.Content.ReadAsStringAsync();

            System.IO.File.WriteAllText("food_data.json", jsonResponse); // lưu nội dung nhận được vào file food_data.json

            var foodResponse = JsonConvert.DeserializeObject<FoodResponse>(jsonResponse);
            foodList = foodResponse.Data;

            listBoxFood.Items.Clear();
            foreach (var food in foodList) // load món ăn lên listbox
            {
                listBoxFood.Items.Add(food);
            }
        }
        else
        {
            MessageBox.Show("Failed to load food data.");
        }
    }
}
}
```

**private void btnRandomFood\_Click(object sender, EventArgs e):** Hàm kích hoạt khi được click của button Random → Random món ăn tùy thuộc vào danh sách đang hiện ( của người dùng tự tạo hay của cộng đồng )

**private void btnAddFood\_Click(object sender, EventArgs e):** Hàm kích hoạt khi được click của button Add Food → mở form AddFoodForm.cs

```
1 reference
private void btnRandomFood_Click(object sender, EventArgs e) // Hàm chọn random món ăn tại tab đang hiện hành
{
    if (foodList != null && foodList.Count > 0) //check foodlist phải khác null và count > 0
    {
        var random = new Random();
        var randomFood = foodList[random.Next(foodList.Count)];
        MessageBox.Show($"Hôm nay ăn: {randomFood.ten_mon_an}\nGiá: {randomFood.gia}\nMô tả: {randomFood.mo_ta}\nĐịa chỉ: {randomFood.dia_chi}");
    }
    else
    {
        MessageBox.Show("Chưa có dữ liệu món ăn.");
    }
}

1 reference
private void btnAddFood_Click(object sender, EventArgs e) // Hàm mở form thêm món ăn
{
    var addFoodForm = new AddFoodForm();
    addFoodForm.Show();
}

4 references
```



**private async Task LoadUserCreatedFoodAsync(int currentUserPage):** Hàm có tác dụng tương tự với hàm **LoadFoodDataAsync()**

Khác ở chỗ hàm này dùng để tải dữ liệu món ăn tự tạo của người đăng nhập → địa chỉ yêu cầu API khác với hàm trên

```
4 references
private async Task LoadUserCreatedFoodAsync(int currentUserPage) // Hàm load món ăn tự tạo
{
    using (HttpClient client = new HttpClient())
    {
        client.DefaultRequestHeaders.Authorization = new System.Net.Http.Headers.AuthenticationHeaderValue(Tokens.TokenType, Tokens.AccessToken);

        var requestBody = new
        {
            current = currentUserPage, // sử dụng page user hiện tại
            pageSize = 5
        };

        var json = JsonConvert.SerializeObject(requestBody);
        var content = new StringContent(json, Encoding.UTF8, "application/json");

        var response = await client.PostAsync("https://nt106.uitiot.vn/api/v1/monan/my-dishes", content); // request url phù hợp

        if (response.IsSuccessStatusCode)
        {
            var jsonResponse = await response.Content.ReadAsStringAsync();
            var foodResponse = JsonConvert.DeserializeObject<FoodResponse>(jsonResponse);
            foodList = foodResponse.Data;

            listBoxFood.Items.Clear();
            foreach (var food in foodList)
            {
                listBoxFood.Items.Add(food);
            }
        }
        else
        {
            MessageBox.Show("Failed to load user-created food data.");
        }
    }
}
```

**private async Task LoadAllFoodAsync(int currentAllPage):** Hàm có tác dụng tương tự với **LoadUserCreatedFoodAsync(int)** nhưng hàm này dùng để rút và tải dữ liệu món ăn từ cả cộng đồng

```
4 references
private async Task LoadAllFoodAsync(int currentAllPage) // Hàm load món ăn all, tương tự với hàm LoadUserCreatedFoodAsync(int)
{
    using (HttpClient client = new HttpClient())
    {
        client.DefaultRequestHeaders.Authorization = new System.Net.Http.Headers.AuthenticationHeaderValue(Tokens.TokenType, Tokens.AccessToken);

        var requestBody = new
        {
            current = currentAllPage, // Use currentAllPage value
            pageSize = 5
        };

        var json = JsonConvert.SerializeObject(requestBody);
        var content = new StringContent(json, Encoding.UTF8, "application/json");

        var response = await client.PostAsync("https://nt106.uitiot.vn/api/v1/monan/all", content); // request url phù hợp

        if (response.IsSuccessStatusCode)
        {
            var jsonResponse = await response.Content.ReadAsStringAsync();
            var foodResponse = JsonConvert.DeserializeObject<FoodResponse>(jsonResponse);
            foodList = foodResponse.Data;

            listBoxFood.Items.Clear();
            foreach (var food in foodList)
            {
                listBoxFood.Items.Add(food);
            }
        }
        else
        {
            MessageBox.Show("Failed to load all food data.");
        }
    }
}
```

**private async void btnNext\_Click(object sender, EventArgs e):** Hàm kích hoạt khi được click của button Next → gửi yêu cầu chuyển trang và truyền đối số++ vào hàm tải dữ liệu tương ứng

**private async void btnPrevious\_Click(object sender, EventArgs e):** Hàm kích hoạt khi được click của button Previous → gửi yêu cầu chuyển trang và truyền đối số-- vào hàm tải dữ liệu tương ứng

```
1 reference
private async void btnNext_Click(object sender, EventArgs e) // hàm chuyển sang trang kế tiếp
{
    if (currentList == "AllFood") // tùy vào load loại tab nào để có thể chọn số trang phù hợp
    {
        currentAllPage++;
        await LoadAllFoodAsync(currentAllPage);
    }
    else if (currentList == "UserCreatedFood")
    {
        currentUserPage++;
        await LoadUserCreatedFoodAsync(currentUserPage);
    }
}

1 reference
private async void btnPrevious_Click(object sender, EventArgs e) // hàm chuyển sang trang trước đó
{
    if (currentList == "AllFood")
    {
        if (currentAllPage > 1)
        {
            currentAllPage--;
            await LoadAllFoodAsync(currentAllPage);
        }
    }
    else if (currentList == "UserCreatedFood")
    {
        if (currentUserPage > 1)
        {
            currentUserPage--;
            await LoadUserCreatedFoodAsync(currentAllPage);
        }
    }
}
```

**private async void btnUserCreatedFood\_Click(object sender, EventArgs e):** Hàm kích hoạt khi được click của button User Created Food → điều hướng và thực thi hàm

**LoadUserCreatedFoodAsync(int)** để load tất cả món ăn tự tạo

**private async void btnAllFood\_Click(object sender, EventArgs e):** Hàm kích hoạt khi được click của button All Food → điều hướng và thực thi hàm

**LoadAllFoodAsync(int)** để load tất cả món ăn của cộng đồng

```
1 reference
private async void btnUserCreatedFood_Click(object sender, EventArgs e) // hàm load load tab món ăn tự tạo
{
    currentUserPage = 1;
    currentList = "UserCreatedFood";
    await LoadUserCreatedFoodAsync(currentUserPage);
}

1 reference
private async void btnAllFood_Click(object sender, EventArgs e) //hàm load tab món ăn all
{
    currentAllPage = 1;
    currentList = "AllFood";
    await LoadAllFoodAsync(currentAllPage);
}
```

**private async void btnDeleteFood\_Click(object sender, EventArgs e):** Hàm kích hoạt khi được click của button Delete Food → Xóa món ăn ( chỉ áp dụng với những món đang được chọn trong ListBox và món được tạo bởi người dùng đang đăng nhập)

- Xác định món ăn được xóa ( chọn món ăn trong ListBox và nhấn button Delete Food)
- Gửi API Request với phương thức HTTP DELETE để xóa món ăn với id được triết xuất
- Thông báo người dùng về toàn bộ thông tin của món được xóa

```
1 reference
private async void btnDeleteFood_Click(object sender, EventArgs e) // hàm xóa món ăn tự tạo
{
    if (listBoxFood.SelectedItem is Food selectedFood && selectedFood.nguoi_dong_gop == User.LogginUser)
    // xét trường hợp món ăn phải được selected và biến global trong namespace hiện tại loginUser phải trùng với nguoi_dong_gop
    {
        using (HttpClient client = new HttpClient())
        {
            client.DefaultRequestHeaders.Authorization = new System.Net.Http.Headers.AuthenticationHeaderValue(Tokens.TokenType, Tokens.AccessToken);

            var response = await client.DeleteAsync($"https://nt186.uitiot.vn/api/v1/monan/{selectedFood.id}"); // gửi request HTTP DELETE cho API kèm theo id của món ăn được chọn

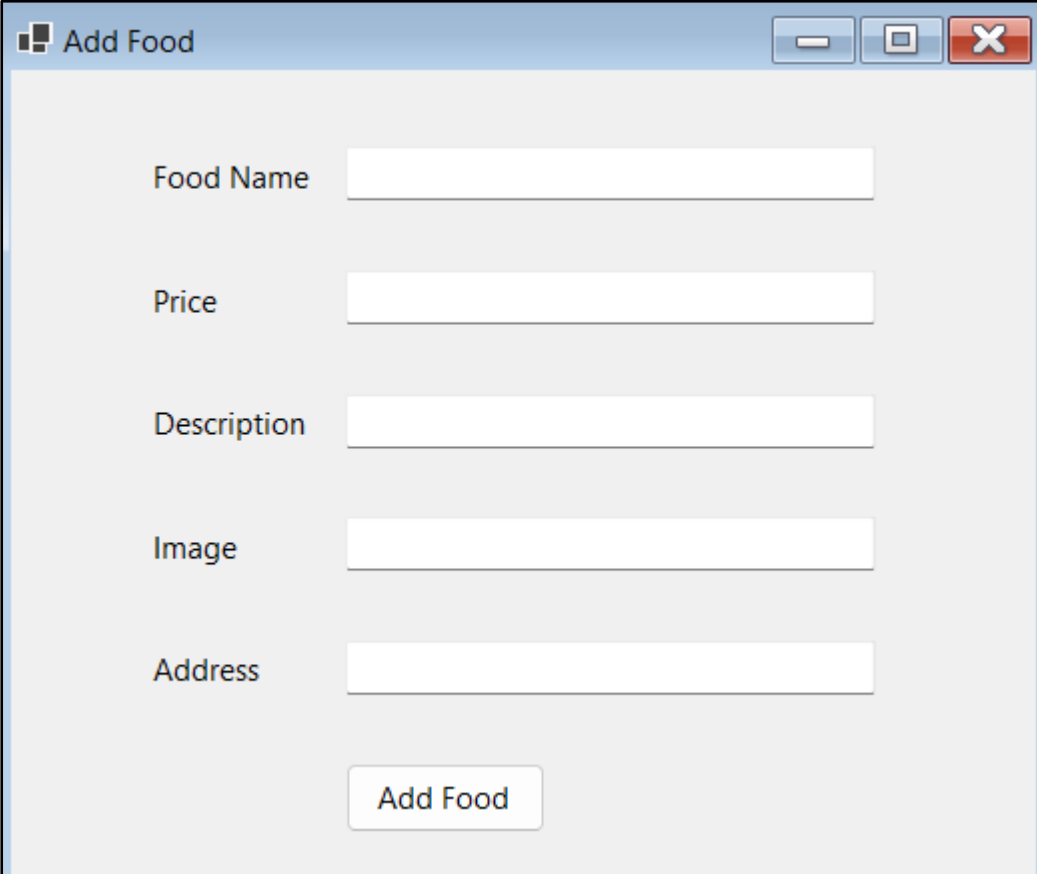
            if (response.IsSuccessStatusCode)
            {
                var jsonResponse = await response.Content.ReadAsStringAsync();
                var deletedFood = JsonConvert.DeserializeObject<Food>(jsonResponse);

                MessageBox.Show($"Deleted Food:\n\n" +
                                $"Tên món ăn: {deletedFood.ten_mon_an}\n" +
                                $"Giá: {deletedFood.gia}\n" +
                                $"Mô tả: {deletedFood.mo_ta}\n" +
                                $"Hình ảnh: {deletedFood.hinh_anh}\n" +
                                $"Địa chỉ: {deletedFood.dia_chi}\n" +
                                $"Người đóng góp: {deletedFood.nguoi_dong_gop}");

                // thông báo toàn bộ thông tin của món ăn mới được xóa

                // tải lại trang
                if (currentList == "AllFood") // nếu xóa tại tab món ăn all
                {
                    await LoadAllFoodAsync(currentAllPage);
                }
                else if (currentList == "UserCreatedFood") // nếu xóa tại tab món ăn tự tạo
                {
                    await LoadUserCreatedFoodAsync(currentUserPage);
                }
            }
            else
            {
                MessageBox.Show("Failed to delete food.");
            }
        }
    }
    else
    {
        MessageBox.Show("You can only delete food items that you have created.");
    }
}
```

Giao diện của form AddFoodForm.cs



The screenshot shows a Windows-style application window titled "Add Food". The window has a standard title bar with minimize, maximize, and close buttons. The main content area is light gray and contains five text input fields, each with a label to its left: "Food Name", "Price", "Description", "Image", and "Address". Below these fields is a single "Add Food" button. The form is designed for adding new food items to a database or system.

**private async void btnAddFood\_Click(object sender, EventArgs e):** Hàm kích hoạt khi được click của button Add Food → Hàm dùng để thêm món ăn vào API

- Xác định thông tin món ăn (theo format mà API yêu cầu)

### Create Monan

AUTHORIZATIONS: >
OAuth2PasswordBearerWithHeader

REQUEST BODY SCHEMA: application/json

ten_mon_an required	string (Ten Mon An)
gia	number (Gia)
mo_ta	string (Mo Ta)
hinh_anh	string (Hinh Anh)
dia_chi	string (Dia Chi)

- Thêm header đã gán Token vào file data
- Thực hiện kết nối với API bằng phương thức POST

```

1 reference
private async void btnAddFood_Click(object sender, EventArgs e) //thu thập thông tin cần thiết
{
    var foodInfo = new
    {
        ten_mon_an = txtFoodName.Text,
        gia = int.Parse(txtPrice.Text),
        mo_ta = txtDescription.Text,
        hinh_anh = txtImage.Text,
        dia_chi = txtAddress.Text,
    };

    string json = JsonConvert.SerializeObject(foodInfo); // convert set thông tin trên thành json
    var data = new StringContent(json, Encoding.UTF8, "application/json"); // json -> string

    using (HttpClient client = new HttpClient())
    {
        // thêm header để được phê duyệt
        client.DefaultRequestHeaders.Authorization = new System.Net.Http.Headers.AuthenticationHeaderValue(Tokens.TokenType, Tokens.AccessToken);

        var response = await client.PostAsync("https://nt106.uitiot.vn/api/v1/monan/add", data);

        if (response.IsSuccessStatusCode)
        {
            MessageBox.Show("Thêm món ăn thành công!");
            this.Close();
        }
        else
        {
            MessageBox.Show("Thêm món ăn thất bại!");
        }
    }
}

```

## 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](https://courses.uit.edu.vn).

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

## HẾT