LẬP TRÌNH MẠNG CĂN BẢN – NT106.O21.ANTN

BÁO CÁO THỰC HÀNH LAB 4

***

Họ và Tên: Dương Phạm Huy Thông

MSSV: 22521431

Link code: tại đây

Bài làm báo cáo những phần được yêu cầu bổ sung sau khi đã báo cáo toàn bộ bài Lab tại lớp trong buổi thực hành số 2.

BÀI LÀM

Bài 1: Lấy và hiển thị nội dung và Header

- Từ URL được nhập vào bởi người dùng, ta thực hiện viết các hàm nhằm lấy thông tin về HTML và Header của một trang web.

```
private string GetHTML(string URL)
{
    try
    {
        // Validate the URL format before making the request
        if (!Uri.IsWellFormedUriString(URL, UriKind.Absolute))
        {
            MessageBox.Show("Invalid URL format. Please enter a valid URL.");
            return "ERROR";
        }

        // Create a request for the URL
        WebRequest request = WebRequest.Create(URL);

        // Get the response
        WebResponse response = request.GetResponse();

        // Get the stream containing content returned by server
        Stream dataStream = response.GetResponseStream();

        // Open the stream using a StreamReader for easy access
        StreamReader reader = new StreamReader(dataStream);

        // Read the content.
        string responseFromServer = reader.ReadToEnd();

        // Close the response
        dataStream.Close();

        return responseFromServer;
    }
    catch (WebException ex)
    {
        MessageBox.Show($"Failed to access URL: {ex.Message}");
        return "ERROR";
    }
    catch (Exception ex)
    {
        MessageBox.Show($"An error occurred: {ex.Message}");
        return "ERROR";
    }
}
```

- Trước tiên, ta cần kiểm tra URL được nhập bởi người dùng đã chính xác với định dạng hay chưa, nếu chưa sẽ hiện thông báo cho người dùng.

- Tạo một **WebRequest** từ URL được nhập bởi người dùng để tạo một yêu cầu cho webserver. Sau đó tạo ra một **WebResponse** để nhận các thông điệp trả về từ **WebRequest** đã được tạo.

- Sau khi nhận được thông điệp từ server, ta thực hiện tạo một luồng ghi dữ liệu bằng đối tượng **StreamReader** và cuối cùng trả về giá trị của nội dung được cung cấp bởi server.

```
private void GetHeader(string URL)
{
    try
    {
        // Create a 'WebRequest' object with the specified url.
        WebRequest myWebRequest = WebRequest.Create(URL);

        // Send the 'WebRequest' and wait for response.
        WebResponse myWebResponse = myWebRequest.GetResponse();

        int index = 1;

        // Display each header and it's key , associated with the response object.
        for (int i = 0; i < myWebResponse.Headers.Count; ++i)
        {
            // Create a new ListViewItem object
            ListViewItem item = new ListViewItem();

            // Set the values for each column
            item.Text = index.ToString();
            index++;
            item.SubItems.Add(new ListViewItem.ListViewSubItem() { Text = myWebResponse.Headers.Keys[i] });
            item.SubItems.Add(new ListViewItem.ListViewSubItem() { Text = myWebResponse.Headers[i] });

            // Add the new ListViewItem to the ListView
            listHeaderInfo.Items.Add(item);
        }

        // Release resources of response object.
        myWebResponse.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Error getting headers: {ex.Message}");
    }
}
```

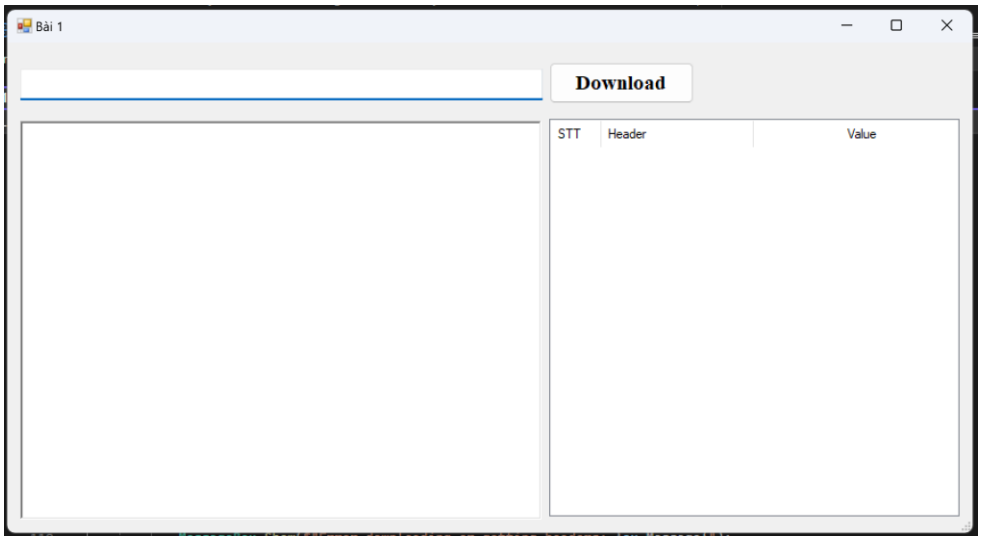
- Với việc lấy thông tin Header mà server phản hồi ta cũng thực hiện tương tự, trong đó ta sẽ bắt đầu với việc tạo một đối tượng **WebRequest** và **WebResponse** tương tự như trên.

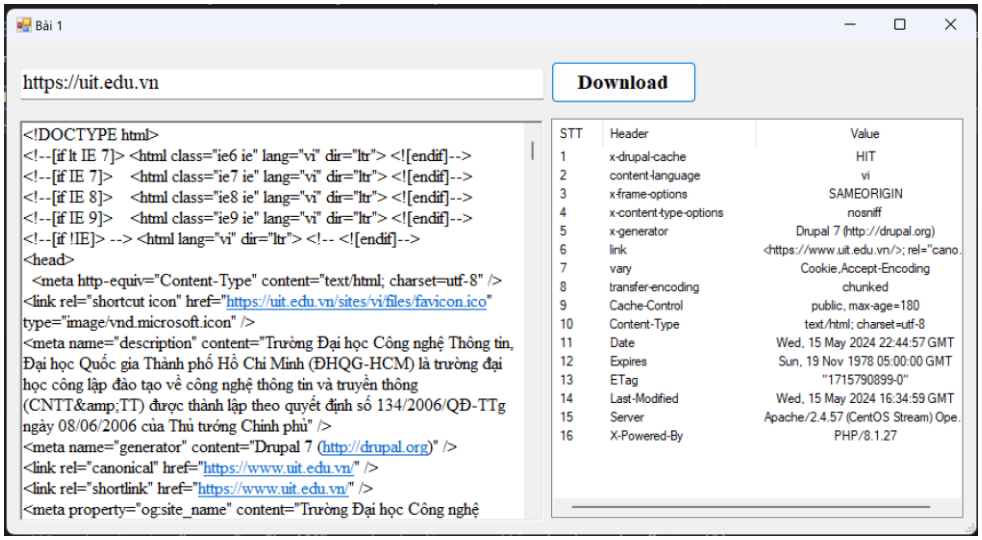
- Sau đó ta duyệt qua tất cả các trường được trả về trong Header thông qua đối tượng **WebResponse** và thực hiện them từng đối tượng trong Header vào **ListView** để hiển thị.

```
private void buttonDownload_Click(object sender, EventArgs e)
{
    // Clear the ListView before download attempt (optional)
    listHeaderInfo.Items.Clear();

    try
    {
        // Download content and get headers (assuming these methods work)
        HTTPContent.Text = GetHTML(textURL.Text);
        GetHeader(textURL.Text);
    }
    catch (Exception ex)
    {
        // Handle exceptions (e.g., display error message to user)
        MessageBox.Show($"Error downloading or getting headers: {ex.Message}");
        buttonDownload.Enabled = false; // Disable button on error
    }
}
```

- Thực hiện sử lý sự kiện **buttonDownload_Click** để chương trình tiến hành lấy nội dung HTML và Header của trang Web, trong đó yêu cầu người dùng nhập vào URL, và truyền URL vào các hàm chức năng sau đó hiển thị kết quả ra màn hình cho người dùng.

Demo:



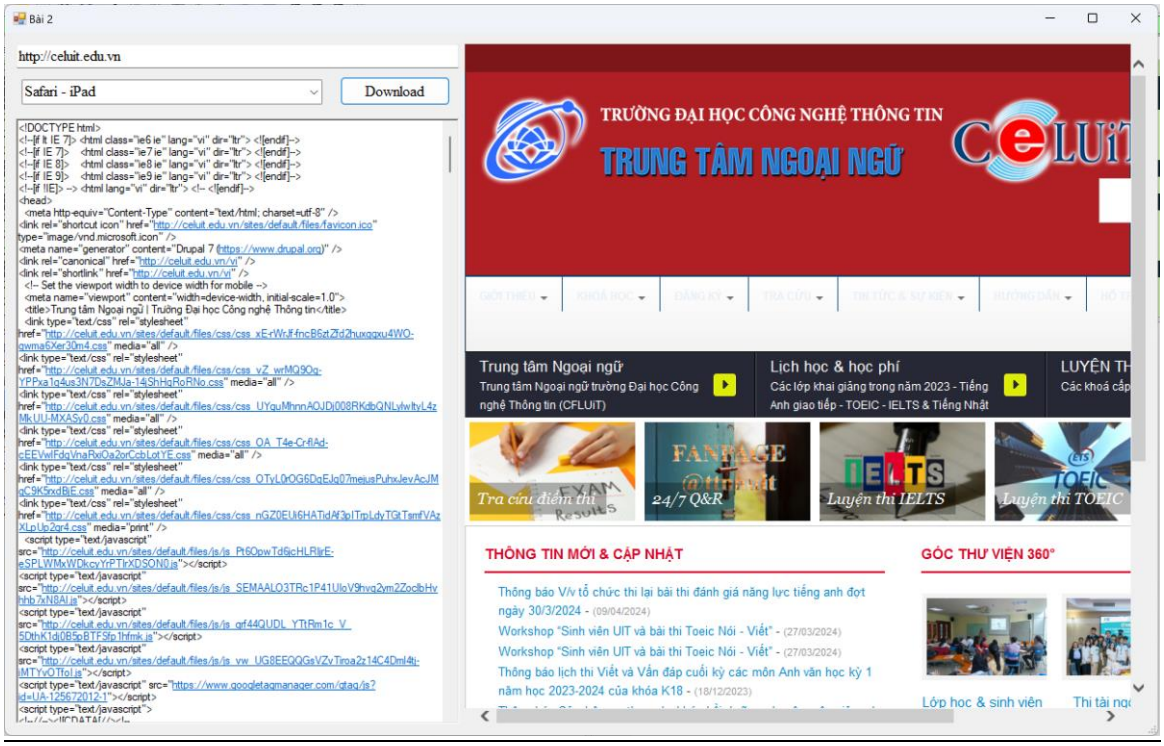


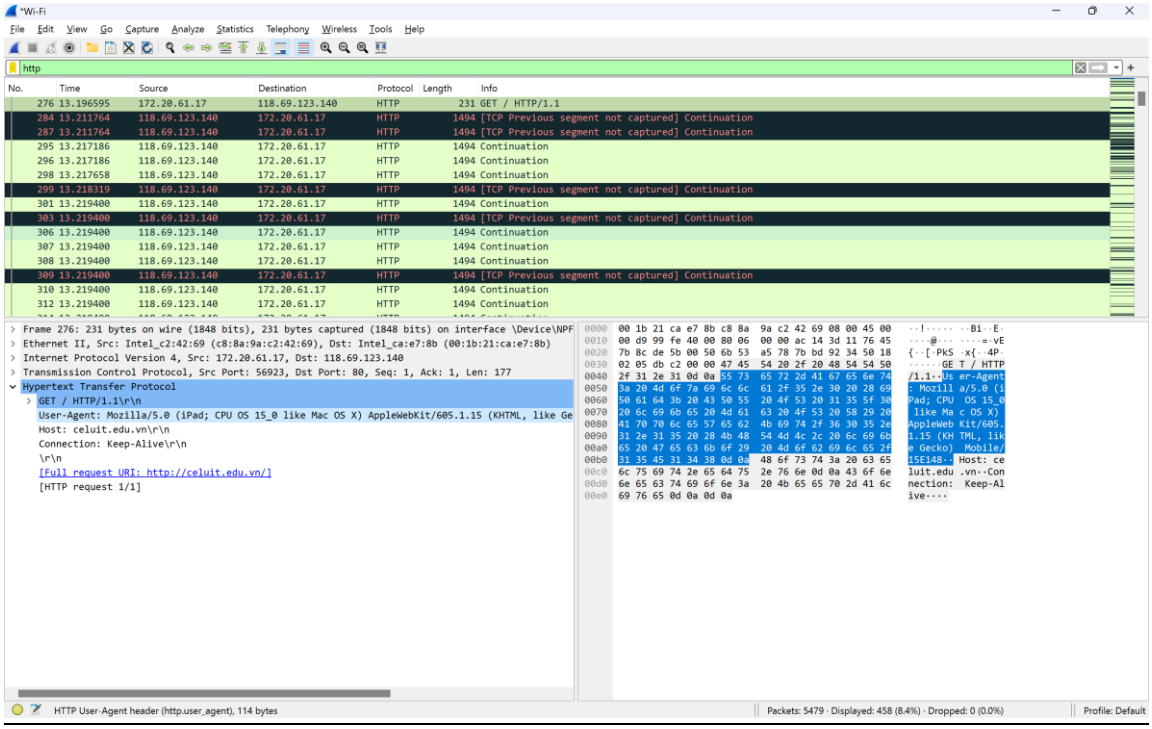
Bài 2: Tiện ích giả lập – Device Emulator

- Yêu cầu người dùng nhập vào một URL bất kỳ, sau đó kiểm tra định dạng của URL đã đúng với mong muốn hay không. Nếu đúng sẽ thực hiện tạo một yêu cầu HTTP đối với webserver thông qua đối tượng **HttpWebRequest**.

- Trước khi gửi thông điệp ta thực hiện lựa chọn **User-Agent** từ người dùng lựa chọn (cung cấp sẵn một số giả lập thiết bị) và gửi đến webserver, sau đó tương tự như Bài 1 ta thực hiện tạo luồng để ghi nhận lại những nội dung được phản hồi lại từ webserver.

Demo:





Bài 3: Làm việc với API

- Trước tiên, chương trình thu thập các giá trị nhập vào từ hai ô văn bản trên giao diện người dùng (**cityNameText** và **countryCodeText**) để lấy tên thành phố và mã quốc gia.

- Xây dựng URL của **API OpenWeatherMap** bằng cách sử dụng tên thành phố, mã quốc gia và khóa API (apiKey) đã cung cấp. Tham số **units=metric** được sử dụng để yêu cầu dữ liệu nhiệt độ ở đơn vị ***Celsius***.

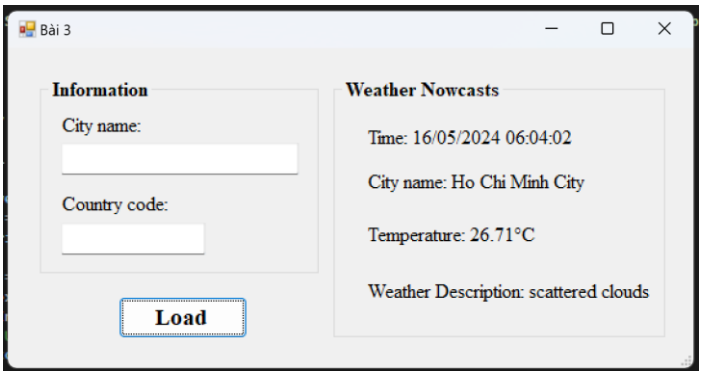
- Sử dụng **HttpClient** để gửi yêu cầu GET đến URL API đã xây dựng. Việc gửi yêu cầu này được thực hiện bất đồng bộ, cho phép ứng dụng tiếp tục thực thi trong khi đợi phản hồi từ máy chủ.

- Nếu yêu cầu thành công, mã đọc nội dung của phản hồi về dưới dạng một chuỗi sử dụng **ReadAsStringAsync()** và sau đó phân tích chuỗi JSON thành các đối tượng **dynamic** bằng cách sử dụng thư viện **Newtonsoft.Json.**

- Sau khi phân tích và lấy dữ liệu thời tiết từ đối tượng JSON, chương trình sử dụng các thông tin này để cập nhật giao diện người dùng bằng cách đặt các giá trị vào các đối tượng hiển thị thông tin về thành phố, nhiệt độ và mô tả thời tiết.

```
string apiUrl = $"http://api.openweathermap.org/data/2.5/weather?q=" +
                $"{city},{countryCode}&appid={apiKey}&units=metric"; // Add API URL

using (HttpClient client = new HttpClient())
{
    try
    {
        HttpResponseMessage response = await client.GetAsync(apiUrl);

        if (!response.IsSuccessStatusCode)
        {
            if (response.StatusCode == System.Net.HttpStatusCode.NotFound)
            {
                MessageBox.Show("City not found. Please enter a valid city name.", "Error", MessageBoxButtons.
            }
            else
            {
                MessageBox.Show($"Error getting weather data: {response.ReasonPhrase}", "Error", MessageBoxBut
            }
            return;
        }

        string responseBody = await response.Content.ReadAsStringAsync();

        dynamic weatherData = Newtonsoft.Json.JsonConvert.DeserializeObject(responseBody);

        string cityName = weatherData.name;
        string temperature = weatherData.main.temp;
        string weatherDescription = weatherData.weather[0].description;

        cityNameInfor.Text = "City name: " + cityName;
        temperatureInfor.Text = "Temperature: " + temperature + "°C";
        weatherDescriptionInfor.Text = "Weather Description: " + weatherDescription;
        // Get the current UTC time
        DateTime dataTimeUtc = DateTime.UtcNow;

        // Convert UTC time to local time using system's default time zone
        DateTime dataTimeLocal = dataTimeUtc.ToLocalTime();

        // Display the local time
        timeInfor.Text = "Time: " + dataTimeLocal.ToString("dd/MM/yyyy HH:mm:ss");

    }
```

Demo:



Bài 4: Trình duyệt cơ bản

Với bài sau, ta thực hiện 3 yêu cầu:

a. Xem nội dung Website (Render HTML)

```
private void Go_btn_Click(object sender, EventArgs e)
{
    string url = URLtxt.Text;

    if (Uri.IsWellFormedUriString(url, UriKind.Absolute))
    {
        try
        {
            // Thêm tiền tố "http://" nếu thiếu
            if (!url.StartsWith("https://"))
            {
                url = "https://" + url;
            }

            // Render trang web bằng WebBrowser control
            webBrowser.Navigate(new Uri(url));
        }
        catch (Exception ex)
        {
            // Hiển thị thông báo lỗi nếu không thể điều hướng
            MessageBox.Show("Không thể điều hướng đến URL: " + ex.Message, "Lỗi", MessageBoxButtons.OK, MessageBo
        }
    }
    else
    {
        // Hiển thị thông báo lỗi nếu URL không hợp lệ
        MessageBox.Show("URL không hợp lệ. Vui lòng nhập URL đúng định dạng.", "Lỗi", MessageBoxButtons.OK, Messa
    }
}
```

- Từ URL được nhập vào bởi người dùng, ta thực hiện kiểm tra định dạng của URL, nếu không có vấn đề với URL ta sẽ thực hiện render trang HTML thông qua phương thức **Navigate** của **WebBrowser.**

b. Download Source(Lưu trữ mã nguồn)

```
private async void Download_btn_Click(object sender, EventArgs e)
{
    string url = URLtxt.Text.Trim();
    string destinationFolder = AppDomain.CurrentDomain.BaseDirectory;

    try
    {
        await DownloadWebsiteAsync(url);
        MessageBox.Show("Tải trang web và các tài nguyên thành công!", "Thông báo", MessageBoxButtons.OK, Message
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Lỗi khi tải trang web: {ex.Message}", "Lỗi", MessageBoxButtons.OK, MessageBoxIcon.Error
    }
}
```

- Tạo một thư mục tạm thời (**src**) trong thư mục hiện tại để lưu trữ tất cả các tài nguyên được tải xuống từ trang web. Sử dụng **HttpClient**, phương thức này tải nội dung HTML của trang web từ URL đã cung cấp.

- Nội dung HTML được lưu vào tệp **index.html** trong thư mục **src**. Sử dụng **HtmlAgilityPack**, phương thức này phân tích HTML để tìm và tải xuống các tài nguyên như hình ảnh, CSS và JavaScript. Nó duyệt qua các thẻ HTML quan trọng

như **<img>, <link>, <script>** và tải các tài nguyên từ các thuộc tính như **src** hoặc **href**.

```
1 reference
public static async Task DownloadWebsiteAsync(string url)
{
    if (!Uri.IsWellFormedUriString(url, UriKind.Absolute))
    {
        throw new ArgumentException("URL không hợp lệ.");
    }

    // Thư mục đích là thư mục 'src' trong thư mục hiện tại của ứng dụng
    string currentDirectory = AppDomain.CurrentDomain.BaseDirectory;
    string destinationFolder = Path.Combine(currentDirectory, "src");

    // Xóa tất cả các tệp và thư mục trong 'src' nếu tồn tại
    if (Directory.Exists(destinationFolder))
    {
        Directory.Delete(destinationFolder, true);
    }

    // Tạo lại thư mục 'src'
    Directory.CreateDirectory(destinationFolder);

    // Tải HTML của trang web
    string html = await client.GetStringAsync(url);

    // Lưu trang HTML vào file
    string htmlFilePath = Path.Combine(destinationFolder, "index.html");
    await WriteTextToFileAsync(htmlFilePath, html);

    // Phân tích HTML để tìm các tài nguyên
    HtmlAgilityPack.HtmlDocument htmlDoc = new HtmlAgilityPack.HtmlDocument();
    htmlDoc.LoadHtml(html);

    // Tải và lưu các tài nguyên (hình ảnh, CSS, JS, ...)
    await DownloadResourcesAsync(htmlDoc, url, destinationFolder);
}
```

- **DownloadResourcesAsync** được sử dụng để tải và lưu trữ các tài nguyên (như hình ảnh, CSS, JavaScript) từ trang web đã được tải về. Nó sử dụng thư viện HttpClient để thực hiện các yêu cầu mạng và lưu trữ các tài nguyên tải xuống vào thư mục đích.

- Phương thức này duyệt qua các thẻ HTML quan trọng như **<img>, <link>, <script>** để tìm các tài nguyên cần tải xuống.

- Với mỗi thẻ HTML tìm thấy, nó lấy giá trị của thuộc tính tương ứng (như **src** cho **<img>**, **href** cho **<link>, src** cho **<script>**).

- Nếu thuộc tính này tồn tại, nó xây dựng URL tuyệt đối bằng cách kết hợp **baseUrl** và giá trị thuộc tính, sau đó gọi phương thức **DownloadResourceAsync** để tải và lưu trữ tài nguyên.

```
private static async Task DownloadResourcesAsync(HtmlAgilityPack.HtmlDocument htmlDoc, string baseUrl, string destinationFolder)
{
    // Các thẻ HTML mà chúng ta quan tâm (img, link, script, ...)
    string[] tags = { "img", "link", "script" };

    foreach (string tag in tags)
    {
        HtmlNodeCollection nodes = htmlDoc.DocumentNode.SelectNodes($"//{tag}");
        if (nodes != null)
        {
            foreach (HtmlNode node in nodes)
            {
                string attribute = GetResourceAttribute(tag);

                if (node.Attributes.Contains(attribute))
                {
                    string resourceUrl = node.Attributes[attribute].Value;
                    string absoluteUrl = new Uri(new Uri(baseUrl), resourceUrl).AbsoluteUri;

                    // Tải và lưu tài nguyên
                    await DownloadResourceAsync(absoluteUrl, destinationFolder);
                }
            }
        }
    }
}
```

- **DownloadResourceAsync** tải một tài nguyên từ URL và lưu trữ nó vào thư mục đích. Phương thức này sử dụng **HttpClient** để tải nội dung của tài nguyên dưới dạng mảng byte. Sau đó, nó xác định tên tệp và loại tài nguyên (dựa trên phần mở rộng của tên tệp). Nó xác định thư mục đích dựa trên loại tài nguyên và tạo thư mục nếu chưa tồn tại. Cuối cùng, nó lưu trữ tài nguyên vào thư mục đích bằng cách gọi phương thức **WriteBytesToFileAsync**.

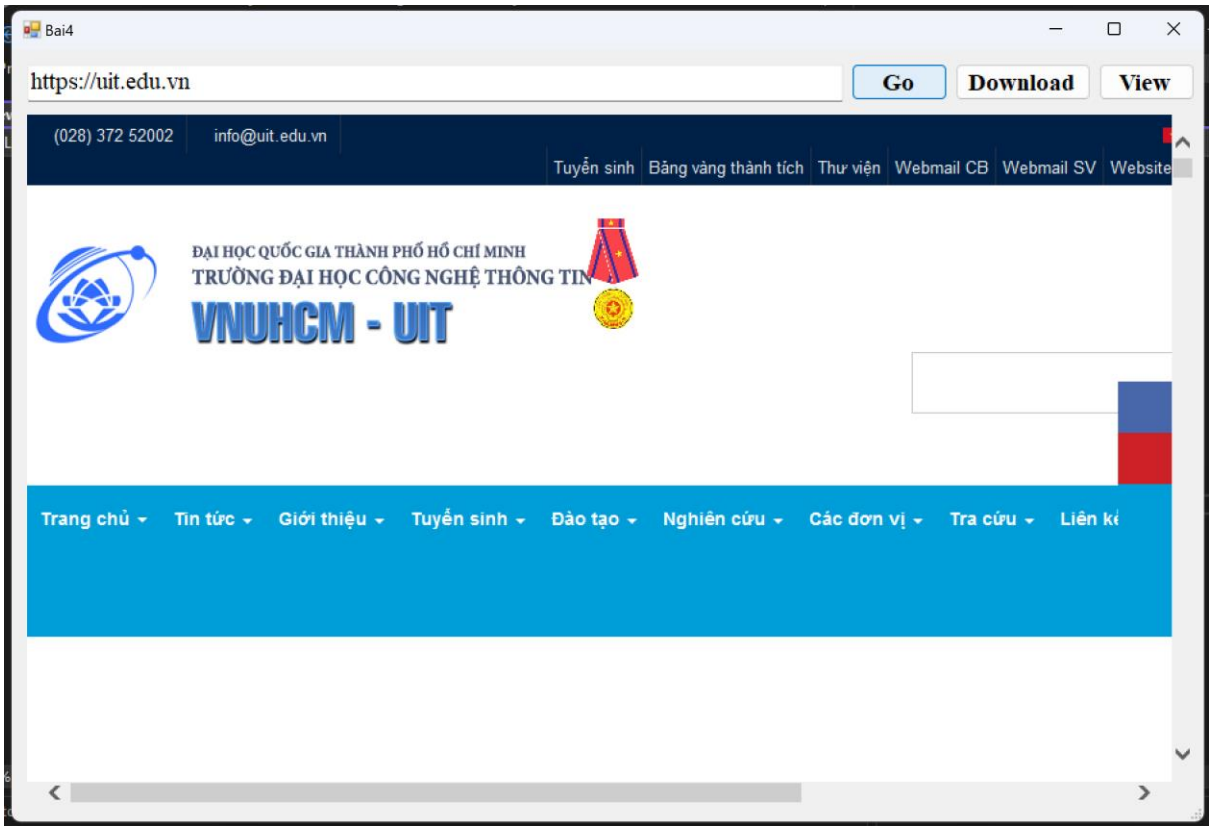
```
private static async Task DownloadResourceAsync(string resourceUrl, string destinationFolder)
{
    try
    {
        // Tải tài nguyên
        byte[] data = await client.GetByteArrayAsync(resourceUrl);

        // Xác định đường dẫn lưu trữ tài nguyên
        Uri uri = new Uri(resourceUrl);
        string fileName = Path.GetFileName(uri.LocalPath);

        // Xác định loại tài nguyên dựa trên đuôi mở rộng của tệp
        string fileExtension = Path.GetExtension(fileName).ToLowerInvariant();
        string resourceTypeFolder = GetResourceTypeFolder(fileExtension);

        // Tạo đường dẫn thư mục cho loại tài nguyên nếu chưa tồn tại
        string resourceTypeFolderPath = Path.Combine(destinationFolder, resourceTypeFolder);
        Directory.CreateDirectory(resourceTypeFolderPath);

        // Lưu tài nguyên vào thư mục tương ứng
        string resourcePath = Path.Combine(resourceTypeFolderPath, fileName);
        await WriteBytesToFileAsync(resourcePath, data);
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Không thể tải tài nguyên {resourceUrl}: {ex.Message}");
    }
}
```

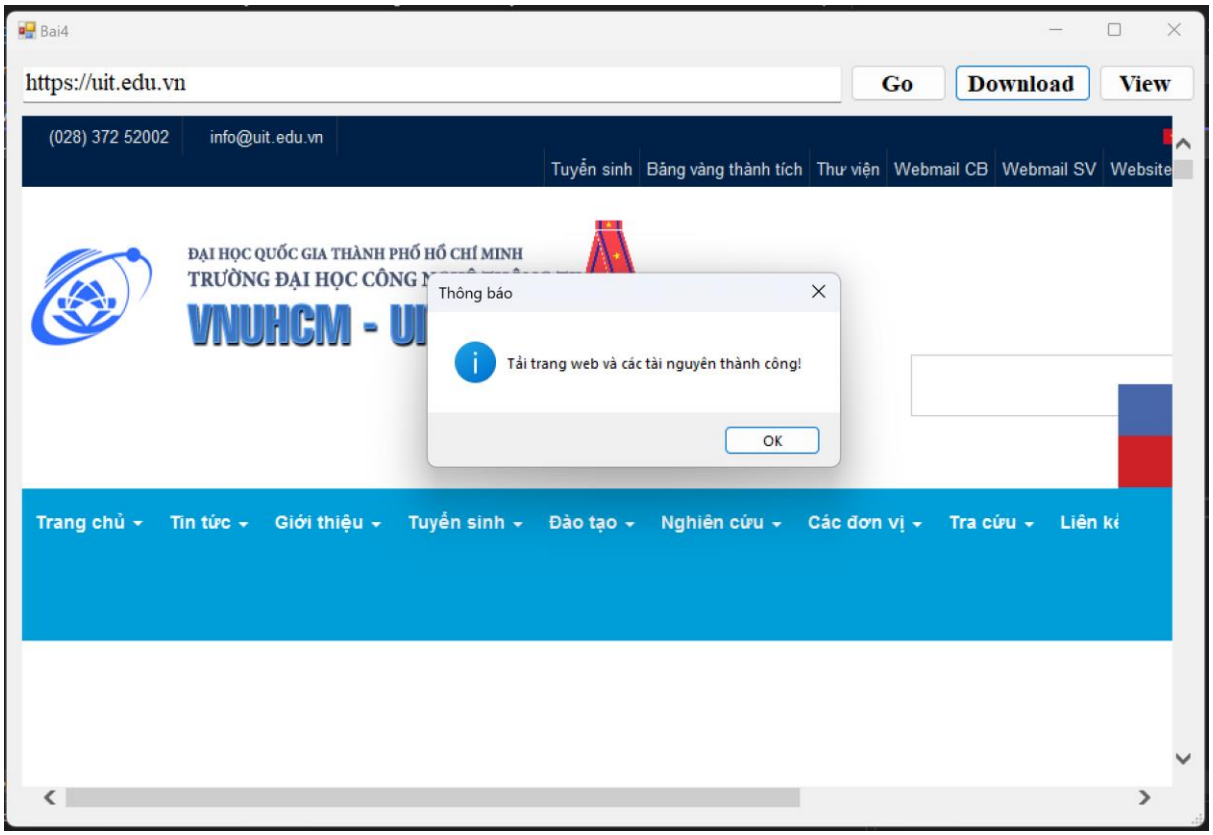
c. Xem Source (Xem mã nguồn)

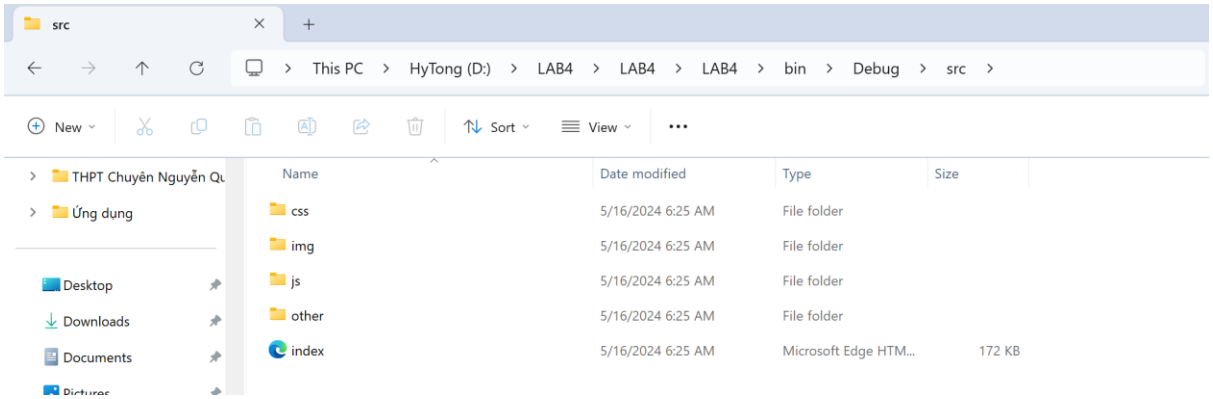
- Lấy ý tưởng từ Bài 1 thực hiện yêu cầu này

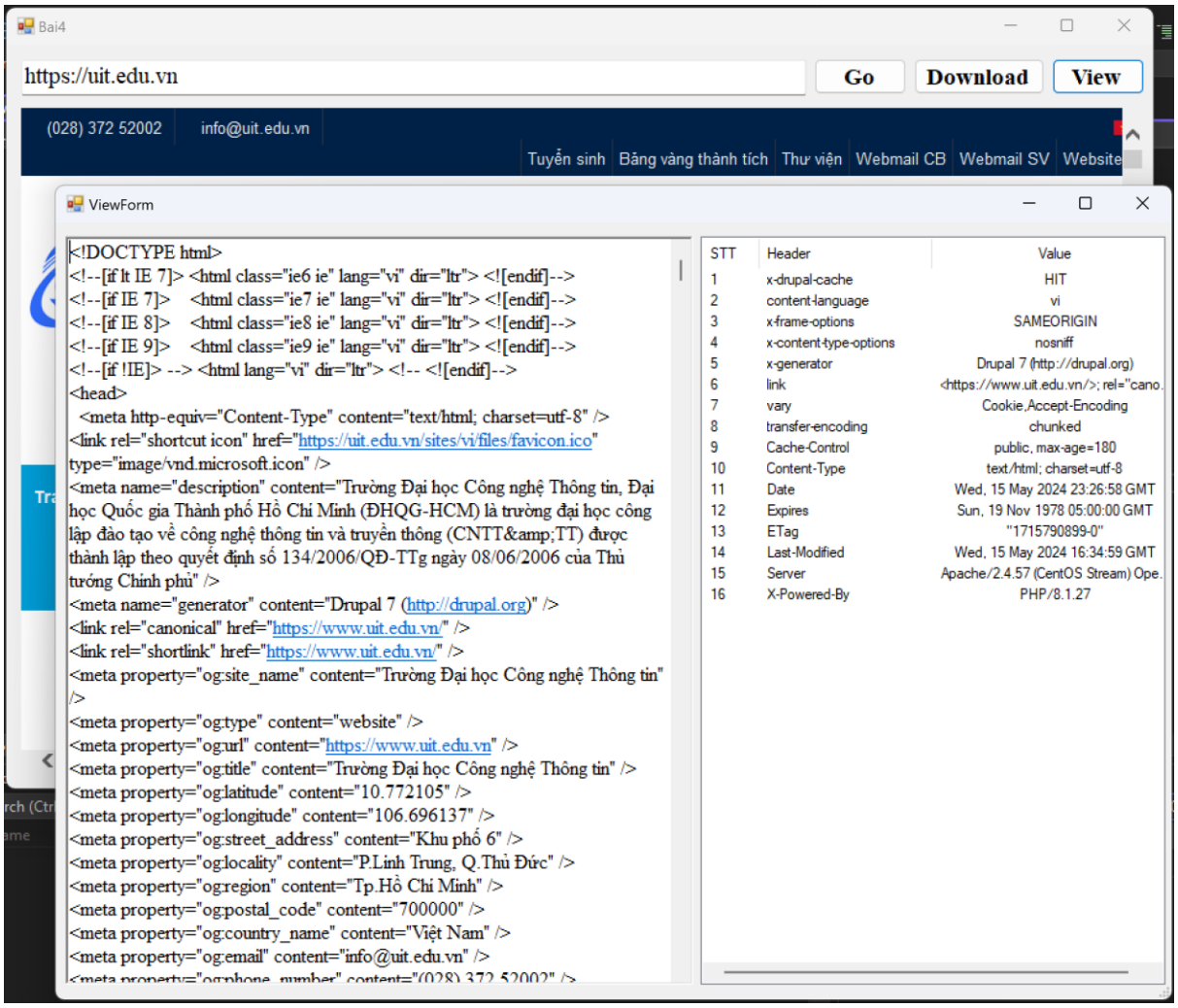
```
1 reference
private void ViewForm_Load(object sender, EventArgs e)
{
    // Clear the ListView before download attempt (optional)
    listHeaderInfo.Items.Clear();

    try
    {
        // Download content and get headers (assuming these methods work)
        HTTPContent.Text = GetHTML(url);
        GetHeader(url);
    }
    catch (Exception ex)
    {
        // Handle exceptions (e.g., display error message to user)
        MessageBox.Show($"Error downloading or getting headers: {ex.Message}");
    }
}
```

Demo:



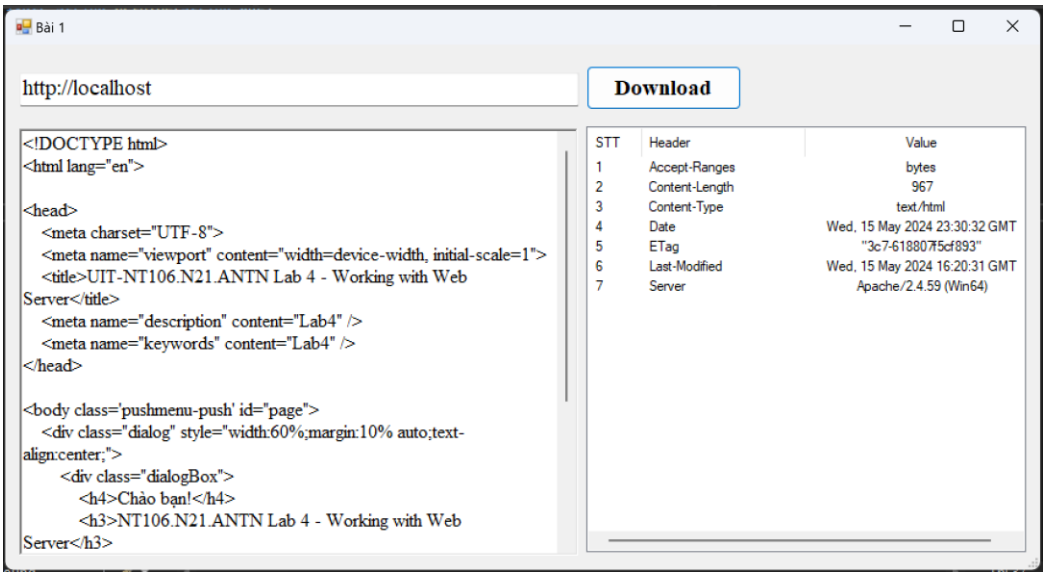

Bai4
https://uit.edu.vn
Go
Download
View
(028) 372 52002
info@uit.edu.vn
Tuyển sinh
Bảng vàng thành tích
Thư viện
Webmail CB
Webmail SV
Website
ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG
VNUHCM - UI
Thông báo
Tải trang web và các tài nguyên thành công!
OK
Trang chủ
Tin tức
Giới thiệu
Tuyển sinh
Đào tạo
Nghiên cứu
Các đơn vị
Tra cứu
Liên kế


src
This PC
HyTong (D:)
LAB4
LAB4
LAB4
bin
Debug
src
New
Sort
View
THPT Chuyên Nguyễn Qu
Ứng dụng
Desktop
Downloads
Documents
Name
Date modified
Type
Size
css
5/16/2024 6:25 AM
File folder
img
5/16/2024 6:25 AM
File folder
js
5/16/2024 6:25 AM
File folder
other
5/16/2024 6:25 AM
File folder
index
5/16/2024 6:25 AM
Microsoft Edge HTM...
172 KB



Bài 1: Lấy và hiển thị nội dung và Header (Mở rộng)

- Sử dụng Apache làm webserver và khi chưa cấu hình thông tin sẽ hiển thị như sau:



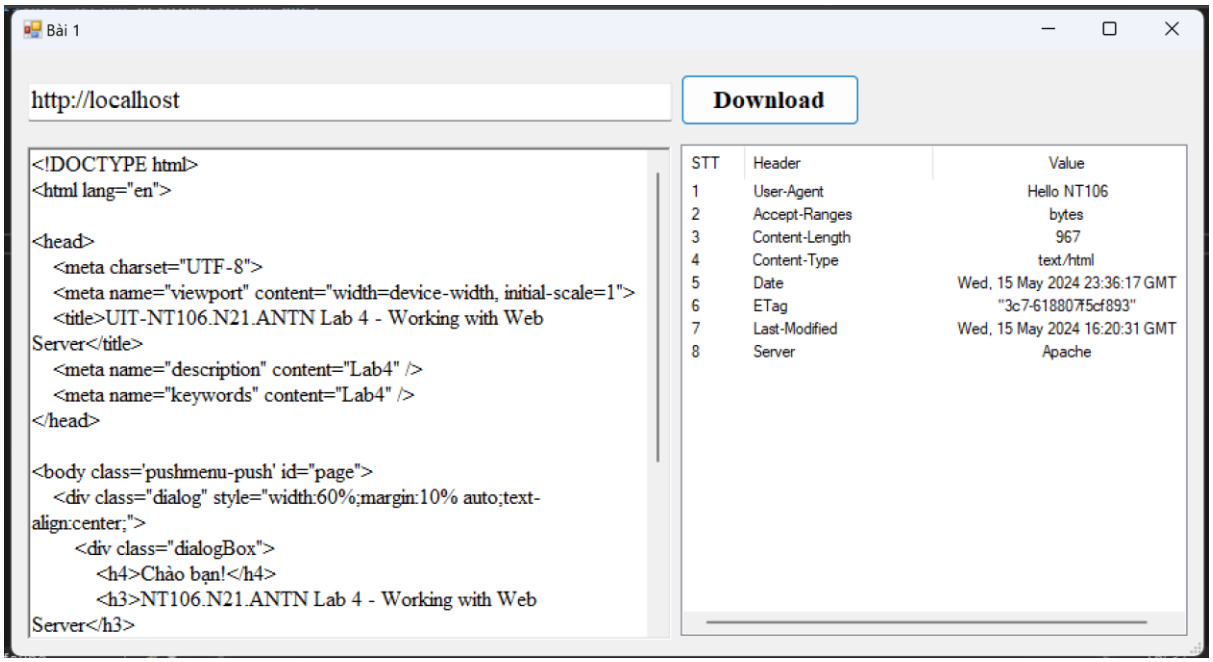
- Sau đó thực hiện cấu hình:

+ Kích hoạt: *LoadModule headers_module modules/mod_headers.so*

+ Thông 2 trường thông tin: *ServerTokens Prod* và *ServerSignature Off* nhằm chỉ định rằng máy chủ sẽ chỉ hiển thị **"ProductOnly" (Prod)** trong các trường **Server response header** thay vì hiển thị phiên bản cụ thể của Apache và chỉ thị để tắt việc hiển thị chữ ký (signature) của máy chủ trong các trang lỗi do máy chủ tạo ra (ví dụ: lỗi 404). Chỉ thị này sẽ không hiển thị bất kỳ thông tin cụ thể nào về máy chủ.

+ Thông thông tin làm giả **User-Agent**: *Header always set User-Agent "Hello NT106"*

- Sau đó thực hiện khởi động lại **apache** để các thay đổi được ghi nhận.



--Hết--