****

|  |  |
| --- | --- |
| **Họ và tên** | **MSSV** |
| Nguyễn Thanh Tùng | 23521744 |
| Đỗ Đức Minh Triết | 23521650 |
| Nguyễn Quốc Việt | 23521782 |

GVHD: **Phan Trung Phát**

Lớp: **NT106.P13.1**

4

**Working with Web Server in C#**

**ĐÁNH GIÁ KHÁC (\*):**

|  |  |
| --- | --- |
| **Nội dung** | **Kết quả** |
| Tổng thời gian thực hiện bài thực hành trung bình (1) | 5 ngày |
| Link Video thực hiện (2)  *(nếu có)* | [Bai3](https://8gcf1d-my.sharepoint.com/personal/posu29_8gcf1d_onmicrosoft_com/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Fposu29%5F8gcf1d%5Fonmicrosoft%5Fcom%2FDocuments%2FVideo%20TH%20l%E1%BA%ADp%20tr%C3%ACnh%20m%E1%BA%A1ng%2FLab04%2DBai03&ga=1) [Bai4](https://drive.google.com/file/d/1McD7kvQlZdJhFGtrY6ZdOoF96i3ooNJb/view?usp=sharing) [Bai6](https://8gcf1d-my.sharepoint.com/personal/posu29_8gcf1d_onmicrosoft_com/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Fposu29%5F8gcf1d%5Fonmicrosoft%5Fcom%2FDocuments%2FVideo%20TH%20l%E1%BA%ADp%20tr%C3%ACnh%20m%E1%BA%A1ng%2FLab04%2DBai06&ga=1) [Bài7](https://drive.google.com/file/d/1nuB7wMT6RxTNPxWgOvDxsmV4KOBlMksT/view?usp=sharing) |
| Ý kiến (3) *(nếu có)*  + Khó khăn  + Đề xuất … |  |
| Điểm tự đánh giá (4) | 8/10 |
| (\*): phần (1) và (4) bắt buộc thực hiện. | |

# Câu hỏi 3

## 1. Minh chứng:

Nội dung.

## A screen shot of a computer program Description automatically generated

Hình . Code bài 3 phần 1

A screen shot of a computer program

Description automatically generated

Hình . Code bài 3 phần 2

A screen shot of a computer program

Description automatically generated

Hình . Code bài 3 phần 3

A screenshot of a computer program

Description automatically generated

Hình . Code bài 3 phần 4

A screenshot of a computer program

Description automatically generated

Hình . Code bài 3 phần 5

A screenshot of a computer

Description automatically generated

Hình . Form chính bài 3

A screenshot of a computer

Description automatically generated

Hình . Form view source bài 3

A screenshot of a computer program

Description automatically generated

Hình . Code view source bài 3

## 2. Giải thích:

* Khởi tạo **form** chính bài 3:
* Biến **url** được khai báo dùng để **lưu trữ URL** mà người dùng nhập vào.
* Trong hàm **Form3()**, sự kiện **Resize** được gắn với phương thức **Form3\_Resize**, giúp cập nhật kích thước của **WebView** khi người dùng thay đổi kích thước cửa sổ.
* **Form3\_Resize(object sender, EventArgs e)**: Điều chỉnh kích thước **WebView** để khớp với kích thước cửa sổ form.
* Gọi **webShow.EnsureCoreWebView2Async(null)** trong **Form3\_Load**, đảm bảo **WebView** được khởi tạo trước khi sử dụng.
* Khi nhấn nút “**Down files**”, phương thức **button\_files\_Click** thực hiện phương thức **downHTML** với url người dùng nhập vào nếu hợp lệ, **fileurl** là web.**html**.
* Phương thức **downHTML**: Tạo một đối tượng **WebClient**, cho phép gửi yêu cầu tới URL để lấy dữ liệu. Gọi **client.OpenRead(szURL)** để mở và đọc nội dung của trang.

Dùng client.**DownloadFile(szURL, fileurl)** để tải toàn bộ trang **HTML** từ **URL** và lưu vào đường dẫn **fileurl**. Nếu file không tồn tại, nó sẽ được tạo mới; nếu đã có, nội dung sẽ được ghi đè.

* Khi nhấn nút “**Load**” phương thức **button\_load\_Click** thực hiện:
* Lấy chuỗi **URL** từ **textBox\_url** và xóa khoảng trắng thừa.
* Nếu **URL** rỗng (**string.IsNullOrWhiteSpace(input)**), không tiếp tục thực hiện load.
* Kiểm tra input có hợp lệ với **pattern** chuẩn không. Nếu hợp lệ, nó sẽ tạo **URL** đầy đủ với tiền tố **http://**
* Nếu **input** bắt đầu bằng **http**, nó gán trực tiếp vào biến url.
* Nếu input không hợp lệ với pattern chuẩn, nó sẽ tìm kiếm **Google** với từ khóa là nội dung input.
* **webShow.CoreWebView2.Navigate(url)** sẽ điều hướng **WebView** đến URL đã xử lý.
* Khi nhấn nút “**Reload**” phương thức **button\_reload\_Click** thực hiện: Kiểm tra **WebView** và **webShow.CoreWebView2** đã được khởi tạo chưa. Nếu có, nó sẽ gọi **webShow.CoreWebView2.Reload()** để tải lại trang.
* Khi nhấn nút “**Download resources**” phương thức **button\_resources\_Click** thực hiện:
* Kiểm tra url có hợp lệ không. Nếu không, phương thức dừng lại.
* Tải nội dung **HTML** của URL bằng **WebClient** và lưu vào **htmlContent**.
* Dùng **HtmlAgilityPack** để phân tích **HTML** và tìm tất cả các thẻ **<img>.**
* Tạo thư mục "**images**" để chứa hình tải về (nếu chưa có).
* Duyệt qua từng **<img>,** lấy **URL** của ảnh từ thuộc tính src. Nếu **URL** không có http, chuyển đổi sang **URL** đầy đủ bằng **baseUri**.
* Tải ảnh về thư mục "**images**". Nếu xảy ra lỗi, hiển thị **MessageBox** thông báo lỗi.
* Phương thức này chỉ tải được hình ảnh công khai từ url.
* Khi nhấn nút “**View source**” phương thức **button\_view\_Click** thực hiện:
* Kiểm tra url có hợp lệ không. Nếu không, phương thức dùng lại.
* Sử dụng **WebClient** để tải nội dung **HTML** của url.
* Tạo một form **Lab04\_Bai03\_ViewSource**, dùng để hiển thị mã nguồn HTML.
* Gọi phương thức **SetHtmlContent** của form **Lab04\_Bai03\_ViewSource** để truyền mã **HTML** vào form, sau đó hiển thị form.
* Nếu có lỗi khi tải, hiển thị thông báo qua **MessageBox**.

# Câu hỏi 4

## 1. Minh chứng:

A screenshot of a computer

Description automatically generated

Hình . Giao diện design của Main

A screen shot of a computer code

Description automatically generated

Hình . Hàm khởi tạo của chương trình Main

A screen shot of a computer program

Description automatically generated

Hình . Class Movie và List movies

A screen shot of a computer program

Description automatically generatedA screen shot of a computer program

Description automatically generated

Hình . Phần 1 của hàm LoadMovieData()

A screen shot of a computer program

Description automatically generatedA computer screen shot of a program code

Description automatically generated

Hình . Phần 2 của hàm LoadMovieData()

A white rectangular object with black border

Description automatically generated

Hình . Giao diện của UserControl

A screen shot of a computer program

Description automatically generated

Hình . Nội dung của UserControl

A screenshot of a computer

Description automatically generated

Hình . Giao diện của Form Browser

A screen shot of a computer program

Description automatically generated

Hình . Nội dung của Form Browser

## 2. Giải thích:

Hàm **Main()** ở Form Main:

- **Nội dung**: Gọi hàm **InitializeComponent()** và **LoadMovieData().**

Class **Movie** ở Form Main:

- **Nội dung**: Lớp dùng để **khởi tạo** đối tượng các **đối tượng Movie**, nhằm dễ dàng hơn trong việc quản lý.

- **Trình bày**: Tạo các thuộc tính **Title (tên phim)**, **Url (đường dẫn thông tin chi tiết của bộ phim)**, **ImageUrl (Link ảnh của bìa của bộ phim)**. Trong các thuộc tính trên có các hàm **get(), set()** để dễ dàng truy xuất.

List **Movies** ở Form Main:

- **Nội dung**: List bao gồm các đối tượng **Movie.**

Hàm **LoadMovieData()** ở Form Main:

- **Nội dung**: Phương thức **bất đồng bộ (async)** trong C# để tải và hiển thị dữ liệu phim từ trang web **https://betacinemas.vn/phim.htm**. Sử dụng các thư viện như **HtmlAgilityPack** để phân tích **cú pháp HTML** và **Newtonsoft.Json** để lưu trữ dữ liệu dưới dạng **file JSON**. Đồng thời, tải về **hình ảnh** phim và hiển thị chúng trên một giao diện **WinForms** có **ListBox** và **FlowLayoutPanel**.

- **Trình bày**:

+ Phần **khởi tạo và tải trang** web:

* **url**: URL của trang web cần tải.
* **web**: Sử dụng HtmlWeb từ thư viện **HtmlAgilityPack** để tải nội dung **HTML** từ **URL** đã chỉ định.
* **doc**: Tải nội dung **HTML** vào một đối tượng **HtmlDocument**

**+** Phân tích **cú pháp HTML** để tìm dữ liệu phim:

* **movieNodes**: Tìm tất cả các **phần tử <div>** có class là **'col-lg-4'**, chứa thông tin phim.
* **progressBar1**: Thanh tiến trình được **hiển thị** để **theo dõi** quá trình tải dữ liệu.

+ Duyệt qua từng **phần tử** phim và lấy **thông tin**:

* **titleNode**: Lấy **tiêu đề** phim và **đường dẫn URL** từ thẻ **<h3><a>.**
* **imageNode**: Lấy đường dẫn hình ảnh từ thẻ **<img>** bên trong phần tử **<div>** có class **'pi-img-wrapper'**.
* **movies.Add(movie)**: **Thêm phim** vào danh sách **movies**.

+ Lưu **dữ liệu** phim dưới dạng **file JSON**:

* **filepath**: Đường dẫn lưu **file JSON**.
* **File.Exists**: Kiểm tra nếu file đã **tồn tại**.
* **File.WriteAllText**: Ghi dữ liệu phim vào **file JSON** nếu chưa tồn tại.

+ Tải hình ảnh và hiển thị trên **FlowLayoutPanel**:

* **movieItem**: Tạo một **MovieItemControl** để hiển thị thông tin phim.
* **sanitizedTitle**: Xử lý tên phim để làm tên **file** hợp lệ (tìm tất cả các ký tự không phải là chữ cái **(in hoa hoặc thường)** hoặc chữ số).
* **DownloadFileTaskAsync**: Tải ảnh về máy và **lưu** vào thư mục cục bộ.
* **flowLayoutPanel1.Controls.Add(movieItem)**: Thêm mục phim vào **FlowLayoutPanel.**

Lớp **Browser(string url)** ở Form Browser:

- **Nội dung**: Đây là **constructor** (hàm khởi tạo) của lớp Browser, nhận vào một **chuỗi url** là đường dẫn trang web mà người dùng muốn mở (**Browser** browserForm = **new Browser**("https://www.example.com")).

Tắt thông báo **lỗi** của **WebBrowser control**:

- **Nội dung**: Dòng lệnh này **tắt các thông báo lỗi JavaScript** khi trang web đang tải trong **WebBrowser control**.

**- Lý do sử dụng**:

* Khi trang web chứa các đoạn mã **JavaScript** bị lỗi, WebBrowser control sẽ hiển thị các thông báo lỗi bật lên (vì tích hợp **Internet Explorer** khác với các công nghệ web **Chromium** hiện tại), gây khó chịu cho người dùng.
* Việc đặt **ScriptErrorsSuppressed = true** giúp **trải nghiệm người dùng mượt hơn** bằng cách **chặn các thông báo lỗi** này.

- Điều hướng **WebBrowser** đến **URL** được chỉ định:

* **webBrowser1.Navigate(url)**: Lệnh này yêu cầu WebBrowser control **mở trang web** có địa chỉ URL mà bạn đã truyền vào constructor.

Lớp **MovieItemControl** ở Form UserControl:  
- **Nội dung**: Là một **UserControl tùy chỉnh** trong C#. Được thiết kế để **hiển thị thông tin về các bộ phim** trong giao diện người dùng của ứng dụng Windows Forms. Nó bao gồm các thuộc tính như **tiêu đề phim (Title)**, **đường dẫn phim (Link)** và **hình ảnh poster (PosterImage)**, cùng với chức năng cho phép người dùng **mở một trang web** khi nhấn vào đường dẫn.

Thuộc tính **Title**:

- **Nội dung**: Đây là thuộc tính công khai dùng để **lấy và đặt nội dung của MovieTitle**, một Label trên UserControl để hiển thị **tiêu đề phim.**

**- Trình bày:**

* **Getter (get)**: Trả về nội dung của **MovieTitle**.
* **Setter (set)**: Đặt nội dung của **MovieTitle** thành giá trị được cung cấp.

Thuộc tính **Link**:

- **Nội dung**: Thuộc tính này dùng để **lấy và đặt đường dẫn phim**. Nó liên kết với **MovieURI**, có thể là một **Label** hoặc **LinkLabel.**

**- Trình bày:**

* **Getter (get)**: Trả về nội dung của **MovieURI**.
* **Setter (set)**: Đặt nội dung của **MovieURI** thành giá trị được cung cấp.

Thuộc tính **PosterImage:**

**- Nội dung:** Thuộc tính này dùng để **lấy và đặt hình ảnh poster** của phim. Nó liên kết với **Poster**, một **PictureBox** trên **UserControl**.

- **Trình bày**:

* **Getter (get)**: Trả về hình ảnh hiện tại của **Poster**.
* **Setter (set)**: Đặt hình ảnh cho **Poster** từ giá trị được cung cấp.

Sự kiện **Click** cho **MovieURI**:

- **Nội dung**: Đây là **sự kiện click** được gắn với MovieURI

- **Trình bày**:

* Khi người dùng nhấn vào đường dẫn (**MovieURI**), sự kiện này sẽ được kích hoạt.
* **Browser(Link)**: Tạo một đối tượng **Browser mới** (dựa trên lớp **Browser** bạn đã định nghĩa trước đó) và truyền vào **URL** của phim.
* **ShowDialog()**: Hiển thị form **Browser** dưới dạng **hộp thoại** (modal dialog), cho phép người dùng xem **nội dung trang web** mà không cần rời khỏi ứng dụng.

**Link Google Drive:** [**https://drive.google.com/file/d/1McD7kvQlZdJhFGtrY6ZdOoF96i3ooNJb/view?usp=sharing**](https://drive.google.com/file/d/1McD7kvQlZdJhFGtrY6ZdOoF96i3ooNJb/view?usp=sharing)

# Câu hỏi 6

## 1. Minh chứng:

A computer screen shot of text

Description automatically generated

Hình . Code bài 6

A screenshot of a computer

Description automatically generated

Hình . Form bài 6

## 2. Giải thích:

* Khởi tạo form với một đối tượng **HttpClient** là một đối tượng thực hiện các yêu cầu **HTTP** tới BaseAddress là ‘**https://nt106.uitiot.vn**’.
* Khi nhấn button ‘SHOW’ phương thức GetUserInfo sẽ được thực hiện.
* Phương thức **GetUserInfo** thực hiện:
* **string jwt = tbToken.Text**: Lấy token người dùng nhập trong TextBox tbToken và lưu vào biến jwt.
* Thiết lập header: **httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("bearer", jwt)** đặt token vào tiêu đề **Authorization** theo chuẩn "**bearer**" để xác thực yêu cầu HTTP.
* Thực hiện yêu cầu GET: Sử dụng **httpClient.GetAsync("api/v1/user/me")** để gửi yêu cầu **GET** đến **endpoint /api/v1/user/me**.
* Kiểm tra phản hồi với **response.IsSuccessStatusCode**
* Sử dụng **var res = await response**.**Content.ReadAsStringAsync()** để đọc nội dung phản hồi dưới dạng chuỗi.
* Phân tích phản hồi với **JObject.Parse** để chuyển chuỗi **JSON** thành đối tượng **JObject**, cho phép trích xuất dữ liệu.
* Hiển thị thông tin **JSON** sau khi phân tích ra **RichTextBox rtbShow** cho người dùng xem.
* Nếu phản hồi không thành công (**response.IsSuccessStatusCode false**), hiển thị hộp thoại thông báo lỗi với **response.ReasonPhrase**.

# Câu hỏi 7

## 1. Minh chứng:

A screenshot of a computer

Description automatically generated

Hình . Form SignUp

A screenshot of a computer

Description automatically generated

Hình 21 Sign In

A screenshot of a computer

Description automatically generated

Hình 22 Form AddDish

A screenshot of a computer

Description automatically generated

Hình 23 Form HeThong

A screenshot of a computer

Description automatically generated

Hình 24 Sau khi đăng nhập

A screenshot of a computer

Description automatically generated

Hình 25 Món ăn tôi đóng góp

A screenshot of a computer

Description automatically generated

Hình 26 Code form SignUp (1)

A screenshot of a computer

Description automatically generated

Hình 27 Code form SignUp (2)

A screenshot of a computer

Description automatically generated

Hình 28 Code form SignIn (1)

A screenshot of a computer

Description automatically generated

Hình 29 Code form SignIn (2)

A screenshot of a computer

Description automatically generated

Hình 30 Code form AddDish

A screenshot of a computer

Description automatically generated

Hình 31 Code form HeThong (1)

A screenshot of a computer

Description automatically generated

Hình 32 Code form HeThong (2)

A screenshot of a computer

Description automatically generated

Hình 33 Code form HeThong (3)

A screenshot of a computer

Description automatically generated

Hình 34 Code form HeThong (4)

A screenshot of a computer

Description automatically generated

Hình 35 Code form HeThong (5)

A screenshot of a computer screen

Description automatically generated

Hình 36 Code form HeThong (6)

A screenshot of a computer

Description automatically generated

Hình 37 Code form HeThong (7)

A screenshot of a computer

Description automatically generated

Hình 38 Code form HeThong (8)

A screenshot of a computer

Description automatically generated

Hình 39 Code form HeThong(9)

## 2. Giải thích:

## Đăng ký

* Hàm **btnSubmit\_Click** là hàm xử lý khi người dùng nhấn nút Submit dùng để:
  + Lấy thông tin từ các textbox.
  + Tạo đối tượng chứa thông tin đăng ký của người dùng, sau đó chuyển đổi thành JSON.
  + Gọi hàm **RegisterUser** để gửi yêu cầu đăng ký tới API.
* Hàm **RegisterUser**: xử lý yêu cầu đăng ký bằng cách gửi dữ liệu người dùng dưới dạng JSON đến API và gửi yêu cầu POST đến API với đường dẫn “**api/v1/user/signup**”.

## Đăng nhập

* Dòng 22 để khai báo và cấu hình một đối tượng **HttpClient** tĩnh để gửi yêu cầu HTTP đến một API.
* Hàm **btnLogin\_Click** để xử lý sự kiên khi người dùng nhấn nút Login để thực hiện đăng nhập:
  + Lấy thông tin đăng nhập và mật khẩu từ textbox.
  + Gọi hàm **LoginAsync** với tên đăng nhập và mật khẩu.
* Khai báo biến **accessToken** để lưu token.
* Hàm **LoginAsync** để thực hiện việc gửi yêu cầu đăng nhập đến API:
  + Tạo đối tượng **formData** chứa các trường dữ liệu đăng nhập và gửi yêu cầu POST đến API với đường dẫn “**auth/token**”.
  + Nếu đăng nhập thành công, lấy **access\_token** từ phản hồi của API và lưu vào biến **accessToken**.
  + Hiển thị thông báo đăng nhập thành công, mở form **HeThong** và truyền **token** cho form này, sau đó ẩn form đăng nhập hiện tại.
  + Nếu có lỗi, hiển thị thông tin lỗi từ API hoặc thông báo lỗi mặc định.

## Thêm món ăn:

* Hàm **btnThem\_Click**: lấy thông tin món ăn từ các textbox, kiểm tra tính hợp lệ của dữ liệu và gửi yêu cầu API để thêm món ăn vào cơ sở dữ liệu sau khi đã đổi nội dung yêu cầu thành JSON, gửi đến “**api/v1/monan/add**”.

## Hệ thống hiển thị các món ăn của bài Hôm nay ăn gì

* **HeThong(string token)** : khởi tạo form, lưu token truy cập, khởi tạo danh sách món ăn và đăng ký các sự kiện.
* **InitializeComboBox():** khởi tạo để chọn trang và đăng ký sự kiện thay đổi chỉ số trang.
* **allToolStringMenuItem\_Click**: gọi API để tải tất cả các món ăn từ server.
* **comboBox1\_SelectedIndexChanged**: lấy thông tin trang từ comboBox và tải lại danh sách món ăn cho trang đó.
* **tôiĐóngGópToolStringMenuItem\_Click** : tải danh sách món ăn mà người dùng đã đóng góp.
* **LoadRandomDish(List<Dish> dishList):** chọn ngẫu nhiên một món ăn từ danh sách và hiển thị thông tin món ăn đó trong một form mới).
* **LoadAllDishes:** 
  + Tạo một đối dượng JSON có 2 tham số **current** và **pageSize**, tạo ra một JSON với các tham số này.
  + Đặt **header Authorization** của yêu cầu HTTP với mã thông báo (token) người dùng đã đăng nhập. accessToken là token người dùng đã có từ quá trình đăng nhập.
  + Gửi yêu cầu HTTP POST tới API với địa chỉ “**ap1/v1/monan/all**”.
  + Nếu phản hồi HTTP trả về thành công, dữ liệu chuyển từ định dạng **JSON** thành đối tượng C#, cụ thể dữ liệu trả về được parse thành một đối tượng **Jobject** và lấy danh sách món ăn từ phần “**data**”.
  + Làm mới danh sách món ăn allDishes.Clear()
  + Hiển thị các thông tin trên tableLayoutPanel1.
* **LoadMyDishes**:
  + Tương tự với **LoadALLDishes** với địa chỉ gửi tới API là “**api/v1/monan/my**-**dishes**”.
* **btnXoaMon\_Click**: cho phép người dùng xoá món ăn đã chọn.
* **DeleteDish(string dishId)** gửi yêu cầu API để xoá món ăn đã chọn, dùng **DeleteAsync** tới địa chỉ API “**api/v1/monan/{ID}**”.
* **tableLayoutPanel1\_MouseClick** chọn món ăn khi người dùng nhấp chuột vào danh sách, tính toán dựa trên toạ độ các hàng của tableLayoutPanel1.
* **btnRandomDish\_Click**: chọn món ăn ngẫu nhiên từ danh sách đóng góp của người dùng hoặc từ tất cả các món ăn.
* **LinkLabellogout\_LinkClicked**: đăng xuất người dùng.

**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.

1. Báo cáo

* Nộp file .docx. Tập trung vào nội dung, giải thích.
* Nội dung trình bày bằng Font chữ Cambria hoặc Times New Roman (*tuy nhiên, phải chuyển đổi hết báo cáo này sang 1 font chữ thống nhất*) – cỡ chữ 13. Canh đều (Justify) cho văn bản. Canh giữa (Center) cho ảnh chụp.
* Đặt tên theo định dạng: Mã lớp-LabX\_MSSV1\_MSSV2. (trong đó X là Thứ tự buổi Thực hành).

Ví dụ: NT106.O21.1-Lab01\_25520001\_25520002

* Nếu báo cáo có nhiều file, nén tất cả file vào file .ZIP với cùng tên file báo cáo.
* Không đặt tên đúng định dạng – yêu cầu, sẽ KHÔNG chấm điểm bài thực hành.
* 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**