**PHẦN I. TRÌNH BÀY (LIỆT KÊ) CÁC BƯỚC THỰC HIỆN TẠO WEB GIÁM SÁT THỜI GIAN THỰC**

**Bước 1: Khảo sát hệ thống**

* **Mục tiêu**: Khảo sát và xác định các yêu cầu để xây dựng hệ thống giám sát thời gian thực.
* **Các yêu cầu**:
  + Lắp đặt cảm biến tại các thiết bị cần giám sát (đèn, quạt, TV).
  + Đảm bảo rằng các cảm biến có khả năng gửi dữ liệu thời gian thực về hệ thống qua giao diện API.
* **Các chức năng phần cứng**:
  + Phần cứng có nhiệm vụ ghi nhận trạng thái của thiết bị (bật/tắt) và truyền dữ liệu đến CSDL thông qua API. Mỗi khi trạng thái thay đổi, dữ liệu cần được cập nhật và lưu trữ vào cơ sở dữ liệu để có thể truy vấn lại lịch sử.

**Bước 2: Xây dựng cơ sở dữ liệu và các thủ tục lưu trữ (Stored Procedure)**

* **Mục tiêu**: Xây dựng các bảng và thủ tục để lưu trữ dữ liệu và trả về JSON cho ứng dụng.
* **Thiết kế Cơ sở Dữ Liệu**
* **Bảng Sensor**: Lưu trạng thái và thông tin của cảm biến:

|  |
| --- |
| CREATE TABLE Sensor (  id INT PRIMARY KEY,  name NVARCHAR(50),  typeID INT, -- Loại cảm biến (1: TV, 2: Quạt, 3: Đèn)  status BIT, -- Trạng thái cảm biến (0: Tắt, 1: Bật)  time DATETIME -- Thời gian cập nhật trạng thái  ); |

* **Bảng History**: Lưu lịch sử trạng thái của các thiết bị

|  |
| --- |
| CREATE TABLE History (  id INT PRIMARY KEY,  SensorID INT,  Status BIT,  Time DATETIME  ); |

* Tạo các thủ tục lưu trữ (SP): Thủ tục SP\_API - Quản lý các chức năng get\_status, get\_history, và control, trả về JSON dựa trên từng loại hành động. Khi gọi action ‘status’, thực hiện thủ tục trả về chuỗi json dữ liệu trạng thái các thiết bị trong bảng Sensor,….

|  |
| --- |
| CREATE PROCEDURE SP\_API  @action VARCHAR(50) = 'get\_status',  @idSensor INT = NULL,  @status BIT = NULL  AS  BEGIN  IF @action = 'get\_status'  BEGIN  DECLARE @json NVARCHAR(MAX) = '{"ok":1,"msg":"ok from db"';  SELECT @json += ',"den":[' + STRING\_AGG(CASE WHEN [status] = 1 THEN '1' ELSE '0' END, ',') + ']'  FROM Sensor WHERE TypeID = 3;  SELECT @json += ',"quat":[' + STRING\_AGG(CASE WHEN [status] = 1 THEN '1' ELSE '0' END, ',') + ']'  FROM Sensor WHERE TypeID = 2;  SELECT @json += ',"tv":' + STRING\_AGG(CASE WHEN [status] = 1 THEN '1' ELSE '0' END, ',') + '}'  FROM Sensor WHERE TypeID = 1;  SELECT @json AS json;  END  ELSE IF @action = 'control'  BEGIN  UPDATE Sensor SET status = @status, time = GETDATE() WHERE id = @idSensor;  INSERT INTO History(SensorID, Status, Time) VALUES(@idSensor, @status, GETDATE());  SELECT '{"ok":1,"msg":"control ok"}' AS json;  END  ELSE IF @action = 'get\_history'  BEGIN  SELECT TOP 100 Status, CONVERT(VARCHAR(19), Time, 120) AS Time  FROM History WHERE SensorID = @idSensor ORDER BY Time DESC FOR JSON PATH;  END  END |

**Chức năng thủ tục SP\_API**:

* **get\_status**: Trả về JSON chứa trạng thái của các thiết bị như đèn, quạt, TV.
* **control**: Cập nhật trạng thái thiết bị và lưu lại lịch sử thay đổi.
* **get\_history**: Trả về JSON chứa lịch sử bật/tắt của một thiết bị nhất định.

**Bước 3: Tạo DLL độc lập kết nối với cơ sở dữ liệu**

**Mục tiêu:** Xây dựng thư viện DLL để kết nối và thực thi các thủ tục lưu trữ trong CSDL.

**Các bước thực hiện:**

* **Khởi tạo DLL:** Tạo dự án Class Library Framwork (DLL) trong Visual Studio.
* Phần chương trình giúp thiết lập kết nối cơ sở dữ liệu

|  |
| --- |
| public class SensorService  {  private string connectionString = "Server=YOUR\_SERVER;Database=YOUR\_DB;User Id=YOUR\_USER;Password=YOUR\_PASSWORD;";  public string GetStatus(int idSensor)  {  using (SqlConnection conn = new SqlConnection(connectionString))  {  conn.Open();  using (SqlCommand cmd = new SqlCommand("SP\_API", conn))  {  cmd.CommandType = CommandType.StoredProcedure;  cmd.Parameters.AddWithValue("@action", "get\_status");  cmd.Parameters.AddWithValue("@idSensor", idSensor);  var result = cmd.ExecuteScalar();  return result?.ToString() ?? "{\"ok\":0,\"msg\":\"Lỗi\"}";  }  }  }  } |

* **Biên dịch và sử dụng:** Sau khi xây dựng, thêm DLL này vào các ứng dụng khác bởi vì đây là class độc lập, có thể dùng cho nhiều trường hợp tuỳ vào các dùng của người lập trình

**Bước 4: Tạo API ASP.NET (Phía Sever) để giao tiếp JSON**

**Mục tiêu:** Xây dựng API cho phép client giao tiếp với server bằng JSON.

**Các bước tạo API:**

* Tạo API Controller trong ASP.NET:

|  |
| --- |
| [HttpPost]  public IActionResult GetStatus(int idSensor)  {  string statusJson = sensorService.GetStatus(idSensor); // sensorService là đối tượng gọi DLL  return Content(statusJson, "application/json");  } |

* **Giao tiếp JSON**: Đảm bảo các phương thức trả về JSON bằng cách sử dụng Content hoặc JsonResult để phản hồi cho client.

**Bước 5**: Giao diện HTML, CSS, jQuery để hiển thị kết quả

**Thiết lập giao diện HTML**: Sử dụng id và class cho các thành phần HTML

|  |
| --- |
| <div id="device-status" class="device">Đèn</div>  <button onclick="refreshStatus()">Làm mới trạng thái</button> |

**CSS**: Định dạng giao diện với id và class

|  |
| --- |
| #device-status {  font-size: 18px;  color: green;  }  .device {  padding: 10px;  border: 1px solid #ddd;  } |

**jQuery và AJAX để gửi/nhận JSON từ API**:

|  |
| --- |
| Dùng AJAX để gửi yêu cầu và xử lý JSON phản hồi:  function refreshStatus() {  $.post("api/GetStatus", { idSensor: 1 }, function(data) {  var json = JSON.parse(data);  $("#device-status").text("Đèn: " + (json.status ? "Bật" : "Tắt"));  });  } |

**jQuery-UI và jQuery-confirm**

**jQuery-UI:** Dùng để thêm chức năng kéo/thả (draggable):

$(".device").draggable();

**jQuery-confirm**: Hiển thị hộp thoại xác nhận khi điều khiển thiết bị

|  |
| --- |
| $.confirm({  title: 'Xác nhận',  content: 'Bạn có muốn bật thiết bị này không?',  buttons: {  confirm: function () {  controlDevice(1, 1); // Gửi lệnh bật thiết bị  },  cancel: function () {}  }  }); |

*Với các bước thiết kế, xây dựng cơ sở dữ liệu, tạo DLL, API ASP.NET, và giao diện, chúng ta đã hoàn thành hệ thống giám sát thời gian thực có khả năng tương tác linh hoạt giữa client và server.*

**PHẦN II. LIỆT KÊ CÁC BƯỚC ĐÃ THỰC HIỆN VỚI 3 BÀI TẬP ĐÃ LÀM**

* 1. **Bài tập 1**

**Đề bài: Tính tiền taxi**

**Yêu cầu:**

* Xây dựng chương trình console tính tiền taxi
* Xây dựng một class độc lập
* Xây dựng chương trình winform tính tiền taxi sử dụng class độc lập.
* Xây dựng web tính tiền sử dụng class độc lập.
* **Xây dựng chương trình console tính tiền taxi**

**Bước 1**: Khởi Tạo Dự Án Console

* Tạo một dự án mới kiểu Console Application trong Visual Studio.
* Đặt tên dự án, ví dụ: ConsoleApp222.

**Bước 2**: Định Nghĩa Biến Đầu Vào và Giải Thuật

* Các biến cần dùng:
  + giaMoCua: Giá mở cửa cố định.
  + quangDuong: Quãng đường di chuyển (do người dùng nhập).
  + giaMoiKm: Giá mỗi km di chuyển (do người dùng nhập).
* Giải thuật:
  + Tính tổng tiền thanhTien dựa trên công thức: thanhTien=giaMoCua+(quangDuong×giaMoiKm)\text{thanhTien} = \text{giaMoCua} + (\text{quangDuong} \times \text{giaMoiKm})thanhTien=giaMoCua+(quangDuong×giaMoiKm)
  + Điều kiện: Cả quangDuong và giaMoiKm phải là số dương. Nếu không, hiển thị thông báo lỗi.

**Bước 3**: Viết Mã Chương Trình Chính

* Cấu trúc Main sẽ có nhiệm vụ:
  1. Nhập quãng đường và giá mỗi km từ người dùng.
  2. Kiểm tra tính hợp lệ của đầu vào.
  3. Áp dụng công thức tính tiền.
  4. Hiển thị kết quả hoặc thông báo lỗi.

**Bước 4**: Kiểm Tra và Xử Lý Đầu Vào

* Kiểm tra đầu vào:
  + Sử dụng double.TryParse để chuyển đổi giá trị nhập từ chuỗi (string) sang số (double).
  + Kiểm tra xem quangDuong và giaMoiKm có là số hợp lệ và dương không.
  + Nếu điều kiện không thỏa mãn, hiển thị thông báo lỗi.

**Bước 5**: Tính Toán và Hiển Thị Kết Quả

* Nếu các điều kiện đầu vào hợp lệ:
  + Áp dụng công thức tính tiền: thanhTien = giaMoCua + (quangDuong \* giaMoiKm).
  + Định dạng kết quả với 2 chữ số thập phân để hiển thị.
* Nếu không hợp lệ:
  + In ra thông báo: "Lỗi: Vui lòng nhập số dương cho quãng đường và giá mỗi km."
* **Xây dựng một class độc lập**

**Tạo dự án DLL**

1. Mở Visual Studio, chọn File > New > Project.
2. Chọn Class Library trong mục C# và đặt tên cho dự án, ví dụ: lib\_tinhtienTAXI.
3. Xóa file Class1.cs mặc định và tạo một class mới, đặt tên là lib\_tinhtienTAXI**.**

**Bước 1**: Tạo Class lib\_tinhtienTAXI

* Tạo một file mới, đặt tên là lib\_tinhtienTAXI (nếu trong dự án console thì bạn có thể tạo class này trực tiếp trong cùng project).
* Đảm bảo class này độc lập về mặt dữ liệ và logic.

**Bước 2**: Khai Báo Các Thuộc Tính Cần Thiết

* Trong class lib\_tinhtienTAXI, định nghĩa các thuộc tính hoặc tham số cần thiết để tính tiền taxi:
  + GiaMoCua: Giá mở cửa cố định.
  + GiaMoiKm: Giá mỗi km di chuyển.
* Có thể khai báo các giá trị này như hằng số nếu không thay đổi hoặc thiết lập chúng thông qua constructor để dễ dàng cấu hình khi tạo đối tượng.

**Bước 3**: Xây Dựng Phương Thức Tính Tiền

* Tạo phương thức TinhTien để tính tiền taxi dựa trên quãng đường di chuyển.
* Logic của phương thức:
  + Kiểm tra tính hợp lệ của quãng đường (phải là số dương).
  + Nếu hợp lệ, tính tiền theo công thức: thanhTien=GiaMoCua+(quangDuong×GiaMoiKm)\text{thanhTien} = \text{GiaMoCua} + (\text{quangDuong} \times \text{GiaMoiKm})thanhTien=GiaMoCua+(quangDuong×GiaMoiKm)
  + Nếu không hợp lệ, có thể trả về giá trị lỗi hoặc thông báo lỗi.
* ***Xây dựng chương trình WinForm sử dụng dll độc lập***

Thêm tham chiếu đến DLL lib\_tinhtienTAXI

1. Nhấp chuột phải vào References trong Solution Explorer của TaxiFareApp.
2. Chọn Add Reference... > Browse..., rồi tìm đến vị trí của lib\_tinhtienTAXI.dll đã tạo và thêm nó vào dự án.

**Bước 1**: Thiết kế giao diện

* Thiết kế form với các điều khiển sau:
  + TextBox để nhập Quãng đường.
  + TextBox để nhập Giá mỗi km.
  + Label để hiển thị Giá mở cửa (cố định, ví dụ: 9000 VND).
  + Button để thực hiện tính tiền.
  + Label hoặc TextBox (ReadOnly) để hiển thị kết quả.

**Bước 2**: Viết mã xử lý khi click nút bấm

|  |
| --- |
| using System;  using System.Windows.Forms;  using TaxiFareLibrary;  namespace TaxiFareApp  {  public partial class MainForm : Form  {  private double baseFare = 9000; // Giá mở cửa cố định  public MainForm()  {  InitializeComponent();  }  private void btnCalculateFare\_Click(object sender, EventArgs e)  {  // Lấy dữ liệu từ TextBox  if (double.TryParse(txtDistance.Text, out double distance) && distance > 0 &&  double.TryParse(txtRatePerKm.Text, out double ratePerKm) && ratePerKm > 0)  {  try  {  // Tạo đối tượng TaxiFareCalculator từ DLL  lib\_tinhtienTAXI fareCalculator = new TaxiFareCalculator(baseFare, ratePerKm);  // Tính tiền taxi  double totalFare = fareCalculator.CalculateFare(distance);  // Hiển thị kết quả  lblTotalFare.Text = $"Tổng tiền taxi: {totalFare:0,0} VND";  }  catch (Exception ex)  {  MessageBox.Show(ex.Message, "Lỗi", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  }  else  {  MessageBox.Show("Vui lòng nhập quãng đường và giá mỗi km là số dương.", "Lỗi nhập liệu", MessageBoxButtons.OK, MessageBoxIcon.Warning);  }  }  }  } |

**Bước 4**: Chạy chương trình và kiểm tra

* Chạy ứng dụng WinForm.
* Nhập quãng đường và giá mỗi km, nhấn nút "Tính Tiền" để tính và hiển thị tổng tiền.
* Kiểm tra các trường hợp nhập liệu sai (ví dụ: quãng đường hoặc giá mỗi km âm hoặc không hợp lệ) để đảm bảo thông báo lỗi được hiển thị.
* ***Xây dựng Web sử dụng dll để tính giá tiền taxi***

Bước 1: Tạo dự án WebForm

1. Mở Visual Studio, chọn File > New > Project.
2. Chọn ASP.NET Web Application (.NET Framework) và đặt tên cho dự án, ví dụ: TaxiFareWebApp.
3. Trong cửa sổ New ASP.NET Web Application, chọn Web Forms và nhấn OK để tạo dự án.

**Bước 2**: Thêm tham chiếu đến DLL lib\_tinhtienTAXI

1. Nhấp chuột phải vào References trong Solution Explorer của dự án TaxiFareWebApp.
2. Chọn Add Reference... > Browse... rồi chọn file lib\_tinhtienTAXI.dll mà bạn đã biên dịch trước đó và thêm vào dự án.

**Bước 3**: Thiết kế giao diện WebForm (WebForm.aspx)\

Mở file WebForm.aspx và thêm các điều khiển sau:

* TextBox để nhập Quãng đường (txtDistance).
* TextBox để nhập Giá mỗi km (txtRatePerKm).
* Label để hiển thị Giá mở cửa (cố định, ví dụ: 9000 VND).
* Button để tính tiền (btnCalculate).
* Label hoặc TextBox (ReadOnly) để hiển thị kết quả (lblTotalFare).

**HTML**

|  |
| --- |
| <asp:Label ID="lblHeader" runat="server" Text="Phần mềm tính tiền taxi" Font-Size="Large"></asp:Label>  <br /><br />  <asp:Label ID="lblDistance" runat="server" Text="Nhập quãng đường (km):"></asp:Label>  <asp:TextBox ID="txtDistance" runat="server"></asp:TextBox>  <br /><br />  <asp:Label ID="lblRatePerKm" runat="server" Text="Nhập giá mỗi km:"></asp:Label>  <asp:TextBox ID="txtRatePerKm" runat="server"></asp:TextBox>  <br /><br />  <asp:Label ID="lblBaseFare" runat="server" Text="Giá mở cửa (9000 VND)"></asp:Label>  <br /><br />  <asp:Button ID="btnCalculate" runat="server" Text="Tính Tiền" OnClick="btnCalculate\_Click" />  <br /><br />  <asp:Label ID="lblTotalFare" runat="server" Text="Tổng tiền taxi: " Font-Bold="True"></asp:Label> |

**WebForm.aspx.cs**

|  |
| --- |
| using System;  using lib\_tinhtienTAXI;  namespace TaxiFareWebApp  {  public partial class WebForm : System.Web.UI.Page  {  protected void Page\_Load(object sender, EventArgs e)  {  }  protected void btnCalculate\_Click(object sender, EventArgs e)  {  double baseFare = 9000; // Giá mở cửa cố định  // Kiểm tra và chuyển đổi dữ liệu đầu vào  if (double.TryParse(txtDistance.Text, out double distance) && distance > 0 &&  double.TryParse(txtRatePerKm.Text, out double ratePerKm) && ratePerKm > 0)  {  try  {  // Tạo đối tượng TaxiFareCalculator từ DLL và tính tiền  TaxiFareCalculator fareCalculator = new TaxiFareCalculator(baseFare, ratePerKm);  double totalFare = fareCalculator.CalculateFare(distance);  // Hiển thị kết quả  lblTotalFare.Text = "Tổng tiền taxi: " + totalFare.ToString("0,0") + " VND";  }  catch (Exception ex)  {  lblTotalFare.Text = "Lỗi: " + ex.Message;  }  }  else  {  lblTotalFare.Text = "Lỗi: Vui lòng nhập số dương cho quãng đường và giá mỗi km.";  }  }  }  } |

|  |  |
| --- | --- |
| **Mã QR dẫn tới github for me – bài 1** | **Link github for me** |
|  | https://github.com/thangtran1904/Tinhtientaxi123 |

**2. Bài tập 2**

**Đề bài: Tạo website giám sát vị trí các thành viên trong lớp 57kmt (ở ktx, ở nhà trọ, ở trường, ở chợ, không biết đang ở đâu) . ko cần vị trí gps**

* Sử dụng database SQL Server với cấu trúc phù hợp.
* Giao diện web cho phép cập nhật thay đổi vị trí mỗi thành viên.
* Thống kê dạng bảng sự thay đổi vị trí của mỗi thành viên.
* Báo cáo thời điểm tất cả các thành viên đều có ở trường.
* Sử dụng database SQL Server với cấu trúc phù hợp.

Tạo bảng **ViTriThanhVien**

|  |
| --- |
| CREATE TABLE [dbo].[ViTriThanhVien](  Mã thành viên  NULL, -- NULL, -- Vị trí hiện tại (KTX, Tr [lastUpdate] [datetime] NULL, -- Thời gian cập nhật vị trí  CONSTRAINT [PK\_ViTriThanhVien] PRIMARY KEY CLUSTERED  (  [uid] ASC  ) ON [PRIMARY]  ); |

|  |
| --- |
| Chèn dữ liệu mẫu vào bảng ViTriThanhVien  INSERT INTO [dbo].[ViTriThanhVien] ([uid], [name], [location], [lastUpdate])  VALUES  (N'thang', N'Trần Chiến Thắng', N'KTX', GETDATE()),  (N'thanh', N'Phạm Ngọc Thuận Thanh', N'Trường', GETDATE()); |

Tạo thủ tục **SP\_QuanLyViTri** khi thực thi trả về chuỗi json

|  |
| --- |
| CREATE PROCEDURE [dbo].[sp\_QuanLyViTri]  @action VARCHAR(50), -- Tham số để xác định hành động  @uid VARCHAR(50) = NULL, -- Mã thành viên (chỉ dùng khi action = 'capnhat')  @location NVARCHAR(100) = NULL -- Vị trí mới của thành viên (chỉ dùng khi action = 'capnhat')  AS  BEGIN  -- Kiểm tra giá trị của @action và thực hiện hành động tương ứng  IF @action = 'capnhat'  BEGIN  -- Cập nhật vị trí của thành viên  IF @uid IS NOT NULL AND @location IS NOT NULL  BEGIN  UPDATE [dbo].[ViTriThanhVien]  SET [location] = @location,  [lastUpdate] = GETDATE()  WHERE [uid] = @uid;  IF @@ROWCOUNT > 0  BEGIN  -- Trả về JSON thông báo thành công  SELECT 'Cập nhật vị trí thành công' AS message  FOR JSON PATH, WITHOUT\_ARRAY\_WRAPPER;  END  ELSE  BEGIN  -- Trả về JSON thông báo lỗi  SELECT 'Mã thành viên không tồn tại' AS message  FOR JSON PATH, WITHOUT\_ARRAY\_WRAPPER;  END  END  ELSE  BEGIN  -- Trả về JSON thông báo thiếu tham số  SELECT 'Cần cung cấp đủ mã thành viên và vị trí' AS message  FOR JSON PATH, WITHOUT\_ARRAY\_WRAPPER;  END  END  ELSE IF @action = 'thongke'  BEGIN  -- Thống kê vị trí của tất cả thành viên  SELECT [uid], [name], [location], [lastUpdate]  FROM [dbo].[ViTriThanhVien]  ORDER BY [lastUpdate] DESC  FOR JSON PATH;  END  ELSE IF @action = 'baocao'  BEGIN  -- Báo cáo khi tất cả thành viên ở trường  SELECT CASE  WHEN NOT EXISTS (SELECT \* FROM [dbo].[ViTriThanhVien] WHERE [location] <> N'Trường')  THEN 'Tất cả thành viên ở trường'  ELSE 'Có thành viên chưa ở trường'  END AS Status  FOR JSON PATH, WITHOUT\_ARRAY\_WRAPPER;  END  ELSE  BEGIN  -- Trả về JSON thông báo hành động không hợp lệ  SELECT 'Hành động không hợp lệ' AS message  FOR JSON PATH, WITHOUT\_ARRAY\_WRAPPER;  END  END  GO |

Chuỗi json trả về có dạng:

|  |
| --- |
| [  {  "uid": "thang",  "name": "Trần Chiến Thắng",  "location": "Trường",  "lastUpdate": "2024-11-13T10:00:00"  },  {  "uid": "thanh",  "name": "Phạm Ngọc Thuận Thanh",  "location": "KTX",  "lastUpdate": "2024-11-13T10:05:00"  }  ] |

Ở trong phía sever chúng ta sẽ xử lý dữ liệu để cho có thể đọc và xử lý chuỗi json này.

**Giao diện HTML và CSS**

|  |
| --- |
| **Html**  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8" />  <title>XD UI đơn giản </title>  <script src="js/jquery.min.js"></script>  <script src="js/jquery-ui.min.js"></script>  <link href="js/jquery-ui.min.css" rel="stylesheet" />  <script src="myjs.js"></script>  <link href="mycss.css" rel="stylesheet" />  <script src="js/bootstrap.bundle.min.js"></script>  <link href="js/bootstrap.min.css" rel="stylesheet" />  <script src="js/crypto-js.min.js"></script>  <script src="js/fp.min.js"></script>  <link href="js/jquery-confirm.min.css" rel="stylesheet" />  <script src="js/jquery-confirm.min.js"></script>  <link href="js/jquery-ui.min.css" rel="stylesheet" />  </head>  <body>  <h1>Bt: Xây dựng website giám sát vị trí các thành viên trong lớp 57km</h1>  <div id="Nen">  <div id="Market1" class="Market"></div>  <div id="Kytuc1" class="Kytuc"></div>  <div id="Truonghoc1" class="Truonghoc"></div>  <div id="Tro1" class="Tro"></div>  <div id="Khongcothongtin" class="No"></div>  <span id="user-info"></span>  <button id="btnLogin" class="btn btn-info">Đăng Nhập</button>  <button class="btn btn-danger no-display cmd-logout">Logout</button>  <div id="Loi"></div>  </body>  </html> |

|  |
| --- |
| **CSS**  #Nen {  background-image: url('images/nền.png');  width: 713px;  height: 480px;  }  #Market1 {  width: 215px;  height: 220px;  position: relative;  left: 469px;  top: 27px;  }  .Market.Market\_dangkhongcosv {  background-image: url('images/chợ đang không có sinh viên.png');  }  .Market.Market\_dangcosv {  background-image: url('images/có sinh viên đang ở chợ.png');  }  #Tro1 {  width: 189px;  height: 96px;  position: relative;  left: 488px;  top: -180px;  }  .Tro\_dangkhongcosv {  background-image: url('images/ở trọ đang không có sinh viên.png');  }  .Tro\_dangcosv {  background-image: url('images/ở trọ đang có sinh viên.png');  }  #Truonghoc1 {  width: 310px;  height: 150px;  position: relative;  left: 22px;  top: -74px;  }  .Truonghoc\_dangkhongcosv {  background-image: url('images/ở trường đang không có sinh viên.png');  }  .Truonghoc\_dangcosv {  background-image: url('images/ở trường đang có sinh viên.png');  }  #Kytuc1 {  width: 192px;  height: 154px;  position: relative;  left: 32px;  top: -171px;  }  .Kytuc\_dangkhongcosv {  background-image: url('images/kí túc xá đang không có sinh viên.png');  }  .Kytuc\_dangcosv {  background-image: url('images/kí túc xá đang có sinh viên.png');  }  #Khongcothongtin {  width: 158px;  height: 139px;  position: relative;  left: 283px;  top: -583px;  }  .No {  background-image: url('images/sinh viên.png');  }  .no-display {  display: none;  } |

**myjs.js**

Các bước thực hiện:

Bước 1. Tạo yêu cầu API để lấy trạng thái vị trí thành viên:

* Sử dụng $.post để gửi yêu cầu API và nhận kết quả JSON về vị trí của các thành viên.
* Cập nhật trạng thái của các thành viên trong lớp dựa trên thông tin trả về từ API.

Bước 2. Thực hiện yêu cầu API định kỳ:

* Dùng setInterval để gửi yêu cầu API mỗi giây nhằm cập nhật trạng thái vị trí của các thành viên.

Bước 3. Cập nhật giao diện dựa trên dữ liệu trả về:

* Cập nhật các phần tử trên giao diện (ví dụ: #Market1, #Kytuc1, #Truonghoc1, #Tro1) để thể hiện trạng thái hiện tại của các thành viên.

Bước 4. Hiển thị thông tin khi thành viên ở trường:

* Kiểm tra xem tất cả các thành viên có ở trường hay không và hiển thị kết quả tương ứng.

|  |
| --- |
| $(function () {  'use strict';  const api = 'api.aspx'; // Địa chỉ API  // Hàm gọi API để lấy trạng thái vị trí của các thành viên  function request\_api\_to\_get\_student\_status() {  $.post(api, { action: 'thongke' }, function (data) {  var json = JSON.parse(data);  if (json.ok) {  // Cập nhật giao diện dựa trên vị trí của các thành viên  $("#Market1").removeClass("Market\_dangkhongcosv Market\_dangcosv");  $("#Kytuc1").removeClass("Kytuc\_dangkhongcosv Kytuc\_dangcosv");  $("#Truonghoc1").removeClass("Truonghoc\_dangkhongcosv Truonghoc\_dangcosv");  $("#Tro1").removeClass("Tro\_dangkhongcosv Tro\_dangcosv");  $("#Khongcothongtin").removeClass("No");  if (json.Market1 == 0)  $("#Market1").addClass("Market\_dangkhongcosv");  else  $("#Market1").addClass("Market\_dangcosv");  if (json.Kytuc1 == 0)  $("#Kytuc1").addClass("Kytuc\_dangkhongcosv");  else  $("#Kytuc1").addClass("Kytuc\_dangcosv");  if (json.Truonghoc1 == 0)  $("#Truonghoc1").addClass("Truonghoc\_dangkhongcosv");  else  $("#Truonghoc1").addClass("Truonghoc\_dangcosv");  if (json.Tro1 == 0)  $("#Tro1").addClass("Tro\_dangkhongcosv");  else  $("#Tro1").addClass("Tro\_dangcosv");  if (json.Khongcothongtin == 1)  $("#Khongcothongtin").addClass("No");  else  $("#Khongcothongtin").addClass(" ");  } else {  $('#Loi').html(json.msg);  }  });  }  // Lặp lại yêu cầu API mỗi giây  setInterval(function () {  request\_api\_to\_get\_student\_status();  }, 1000);  // Kiểm tra tất cả thành viên có ở trường hay không  function check\_all\_students\_in\_school() {  $.post(api, { action: 'baocao' }, function (data) {  var json = JSON.parse(data);  if (json.Status === "Tất cả thành viên ở trường") {  $('#Status').html("Tất cả thành viên đều ở trường");  } else {  $('#Status').html("Có thành viên chưa ở trường");  }  });  }  // Kiểm tra khi load trang  check\_all\_students\_in\_school();  }); |

**api.aspx**

* Định nghĩa các lớp dữ liệu cho thành viên, vị trí và lịch sử cập nhật.
* Tạo hàm thực thi stored procedure để lấy dữ liệu từ cơ sở dữ liệu.
* Cập nhật thông tin thành viên và vị trí thông qua API.
* Lấy lịch sử cập nhật từ cơ sở dữ liệu hoặc API.
* Trả về dữ liệu dưới dạng JSON để dễ dàng xử lý ở phía client.

**Bước 1**: Cấu hình kết nối CSDL

|  |
| --- |
| private string cnstr;  public api()  {  // Đọc chuỗi kết nối từ file cấu hình  cnstr = ConfigurationManager.ConnectionStrings["cnstr"].ConnectionString;  } |

**Bước 2**: Định nghĩa lớp PhanHoi

* Tạo lớp để xử lý phản hồi JSON khi có lỗi hoặc thành công.

|  |
| --- |
| class PhanHoi  {  public bool ok;  public string msg;  public int Market1;  public int Kytuc1;  public int Truonghoc1;  public int Tro1;  public int Khongcothongtin;  } |

**Bước 3**:

Xây dựng hàm thực thi SP

|  |
| --- |
| private string ExecuteStoredProc(string storedProcName, Dictionary<string, object> parameters)  {  string result = string.Empty;  using (var conn = new SqlConnection(cnstr))  {  try  {  SqlCommand cmd = new SqlCommand(storedProcName, conn);  cmd.CommandType = CommandType.StoredProcedure;  // Add parameters  foreach (var param in parameters)  {  cmd.Parameters.AddWithValue(param.Key, param.Value);  }  conn.Open();  SqlDataReader reader = cmd.ExecuteReader();    // Read data  while (reader.Read())  {  // Process the data and store in result  // Example: result = reader["ColumnName"].ToString();  }  reader.Close();  }  catch (Exception ex)  {  result = get\_bao\_loi($"Lỗi khi thực thi SP: {ex.Message}");  }  }  return result;  } |

**Bước 4**: Cập nhật thông tin thành viên và vị trí

Khi nhận dữ liệu từ stored procedure, bạn có thể cập nhật thông tin thành viên và vị trí thành viên trong cơ sở dữ liệu hoặc trả về JSON.

|  |
| --- |
| void CapNhatThanhVienVaViTri()  {  Dictionary<string, object> parameters = new Dictionary<string, object>  {  { "@MemberId", memberId }  };  string responseJson = ExecuteStoredProc("sp\_GetMemberInfoAndLocation", parameters);  // Process JSON response and update the member and location details  if (!string.IsNullOrEmpty(responseJson))  {  var memberInfo = JsonConvert.DeserializeObject<ThongTinThanhVien>(responseJson);  var memberLocation = JsonConvert.DeserializeObject<ViTriThanhVien>(responseJson);  // Update the member details in the database or return them in the response  // Example: UpdateDatabase(memberInfo, memberLocation);  }  else  {  bao\_loi("Không có dữ liệu thành viên.");  }  } |

**Bước 5**: Xử lý Lịch Sử Cập Nhật

Lịch sử cập nhật có thể được lưu vào cơ sở dữ liệu hoặc xử lý trực tiếp từ stored procedure. Chúng ta sẽ lấy lịch sử cập nhật và trả về kết quả

|  |
| --- |
| void LayLichSuCapNhat(int memberId)  {  Dictionary<string, object> parameters = new Dictionary<string, object>  {  { "@MemberId", memberId }  };  string responseJson = ExecuteStoredProc("sp\_GetUpdateHistory", parameters);  if (!string.IsNullOrEmpty(responseJson))  {  var historyList = JsonConvert.DeserializeObject<List<LichSuCapNhat>>(responseJson);  // Process or return history data  // Example: return historyList;  }  else  {  bao\_loi("Không có lịch sử cập nhật.");  }  } |

**Bước 6**: Tạo phản hồi JSON và trả về kết quả

Mỗi hàm xử lý sẽ trả về kết quả dưới dạng JSON, đảm bảo dữ liệu luôn được chuẩn hóa và dễ dàng xử lý phía client.

|  |
| --- |
| void bao\_loi(string msg)  {  PhanHoi p = new PhanHoi();  p.ok = false;  p.msg = msg;  string json = JsonConvert.SerializeObject(p);  this.Response.Write(json);  }  void bao\_thanh\_cong(string msg, object data)  {  PhanHoi p = new PhanHoi();  p.ok = true;  p.msg = msg;  p.data = data;  string json = JsonConvert.SerializeObject(p);  this.Response.Write(json);  } |

**Bước 7**: Cập nhật Page\_Load để xử lý các hành động API

Cuối cùng, bạn sẽ cập nhật hàm Page\_Load để lắng nghe và xử lý các yêu cầu từ client như kiểm tra trạng thái thành viên, lấy thông tin thành viên, vị trí và lịch sử cập nhật.

|  |
| --- |
| protected void Page\_Load(object sender, EventArgs e)  {  string action = this.Request.Form["action"];    switch (action)  {  case "get\_member\_info":  int memberId = Convert.ToInt32(this.Request.Form["memberId"]);  CapNhatThanhVienVaViTri(memberId);  break;  case "get\_update\_history":  memberId = Convert.ToInt32(this.Request.Form["memberId"]);  LayLichSuCapNhat(memberId);  break;  default:  bao\_loi("Lỗi action không hợp lệ.");  break;  }  } |

Kết quả:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

|  |  |
| --- | --- |
| **Mã QR dẫn tới github for me – bài 2** | **Link github for me** |
|  | https://github.com/thangtran1904/Giamsatk57kmt |

**3. Bài tập 3**

**Đề bài: Tạo solution gồm các project sau**

1. **Class Library (.NET Framework 2.0): Captcha**

* Input+Output:
  + Input: Nhận vào chuỗi kí tự (khoảng 6 kí tự).
  + Output: Đầu ra là Bitmap chứa 6 kí tự đó. Nhưng phải làm sao để người thì nhìn thấy còn máy thì khó!
* Thuật toán:
  + Nhận chuỗi ngẫu nhiên
  + DLL có sẵn trước khoảng 10 ảnh nền gây lú. Add resource: add 10 ảnh vào resouce là ok.
  + Load 1 ảnh nền bất kỳ ngẫu nhiên (ảnh nền là resource của DLL đó luôn)
  + Vẽ từng ký tự trong chuỗi ngẫu nhiên vào ảnh, mỗi ký tự dùng ngẫu nhiên 1 Font trong 1 tập font xác định trước, mỗi kí tự này còn xoay đi 1 góc ngẫu nhiên (-45 độ đến +45 độ), và mỗi kí tự có 1 size ngâu nhiên, toạ độ (y) của kí tự cũng ngẫu nhiên, còn toạ độ x thì đương nhiên tăng dần để phân biệt thứ tự trước sau của các kí tự, các kí tự có thể có màu ngẫu nhiên và có thể đè lên nhau.
  + Có thể bổ xung thêm các hiệu ứng: làm thế nào đó để chữ nó ngoằn nghèo như sóng (tuỳ ý sáng tạo, tuy nhiên vẫn phải để cho người nhìn thấy chữ)
  + Bổ xung thêm nhiễu vào ảnh: vẽ các đường thẳng, hình tròn, tam giác ở các vị trí ngẫu nhiên, kích thước ngẫu nhiên, màu sắc ngẫu nhiên, có thể đè lên các kí tự để khó nhìn, ngây nhiễu.

1. **Windows Form App (.NET Framework 2.0): Test dispay Captcha**

* Tham chiếu tới DLL trên
* Cho phép nhập text
* Dùng DLL tạo ảnh có chứa text vừa nhập, hiển thị ảnh ra Form (có thể dùng PictureBox, or any other)

1. **ASP.NET Web Application (.NET Framework 2.0): Login with Capcha**

* Mục tiêu: khi login sai 3 lần thì bắt nhập thêm captcha.
* Thuật toán:
* Thông tin user (uid và pwd) lưu trong database SQL Server
* Sử dụng kỹ thuật tạo api.aspx để xử lý các truy vấn từ client gửi lên, kết quả trả lại json hoặc định dạng phù hợp.
* Sử dụng Session để đếm số lần login thất bại (cặp thông tin uid và pwd không khớp với db thì tính là không hợp lệ)
* Nếu đã 3 lần thất bại thì form login sẽ hiển thị ảnh captcha, người dùng cần nhập đúng text trong ảnh captcha thì mới xét tính đúng đắn của uid và pwd => Nếu captcha sai hoặc uid sai hoặc pwd sai thì đều bắt nhập lại với captcha mới.
* Ảnh captcha được sinh ra từ DLL trên
* Text ngẫu nhiên truyền cho captcha cũng được lưu trong session để tiện cho việc đối chiếu với text captcha người dùng nhập vào.
* html dùng thẻ <img src="..." /> sẽ hiển thị được ảnh. thuộc tính src của thẻ img có thể là 1 url, hoặc cũng có thể ở dạng base64
* Sử dụng các thư viện jQuery và jQuery-Confirm để viết code js.

**1. Class Library (.NET Framework 2.0): Captcha**

**Bước 1:** Tạo một Class Library cho Captcha

* Tạo một dự án **Class Library** trong Visual Studio với .NET Framework 2.0.
* Đặt tên cho thư viện là **CaptchaLib**.

**Bước 2:** Thêm các ảnh nền vào Resources

* Thêm khoảng 10 ảnh nền vào tài nguyên của thư viện. Bạn có thể làm điều này bằng cách:
  + Nhấp chuột phải vào thư mục **Resources** trong dự án, chọn **Add > Existing Item** và chọn các ảnh nền mà bạn muốn thêm vào.
  + Chuyển các ảnh vào thư mục Resources và chọn **Embedded Resource** trong thuộc tính **Build Action**.

**Bước 3:** Xây dựng phương thức tạo Captcha

* Tạo một phương thức trong thư viện để nhận vào một chuỗi ký tự ngẫu nhiên, sau đó tạo một ảnh Bitmap chứa chuỗi này với các yêu cầu đã đưa ra.
* Sử dụng Graphics và Font để vẽ các ký tự lên ảnh.
* Các yếu tố cần thiết:
  + **Tạo chuỗi ngẫu nhiên** với 6 ký tự.
  + **Chọn ảnh nền ngẫu nhiên** từ các ảnh đã thêm vào.
  + **Vẽ các ký tự lên ảnh với font, kích thước, màu sắc và góc ngẫu nhiên.**
  + **Thêm hiệu ứng nhiễu** như vẽ các hình tam giác, đường thẳng, hoặc hình tròn vào ảnh.

|  |
| --- |
| **lib\_captcha**  using System;  using System.Drawing;  using System.Drawing.Imaging;  using System.Random;  public class CaptchaGenerator  {  private static Random random = new Random();  public static Bitmap GenerateCaptcha(string captchaText)  {  // Load a random background image  var backgrounds = new string[] { "background1.jpg", "background2.jpg", "background3.jpg" };  string selectedBackground = backgrounds[random.Next(backgrounds.Length)];  Bitmap bitmap = new Bitmap(selectedBackground);  Graphics g = Graphics.FromImage(bitmap);  // Randomize font, color, and position  for (int i = 0; i < captchaText.Length; i++)  {  Font font = new Font("Arial", random.Next(18, 30), FontStyle.Bold);  Brush brush = new SolidBrush(Color.FromArgb(random.Next(256), random.Next(256), random.Next(256)));  int x = 20 + i \* 30; // X is incremented for each character  int y = random.Next(10, 50); // Random Y position  g.RotateTransform(random.Next(-45, 45));  g.DrawString(captchaText[i].ToString(), font, brush, x, y);  g.RotateTransform(-random.Next(-45, 45)); // Reset rotation  }  // Add noise to image (optional)  AddNoise(g, bitmap);  return bitmap;  }  private static void AddNoise(Graphics g, Bitmap bitmap)  {  for (int i = 0; i < 100; i++)  {  int x = random.Next(bitmap.Width);  int y = random.Next(bitmap.Height);  int size = random.Next(5, 15);  Brush brush = new SolidBrush(Color.FromArgb(random.Next(256), random.Next(256), random.Next(256)));  g.FillEllipse(brush, x, y, size, size);  }  }  } |

**Bước 4:** Đóng gói và xuất DLL

* Sau khi hoàn thành các phương thức, biên dịch dự án và xuất DLL để có thể sử dụng trong các ứng dụng khác.

**2. Windows Form App (.NET Framework 2.0): Test Display Captcha**

**Bước 1:** Tạo một Windows Forms Application

* Tạo một dự án **Windows Forms Application** trong Visual Studio, với .NET Framework 2.0.
* Đặt tên cho dự án là **CaptchaTestApp**.

**Bước 2:** Thêm tham chiếu đến DLL

* Thêm tham chiếu đến thư viện **lib\_captcha.dll** (thư viện đã tạo trong phần 1).

**Bước 3:** Tạo giao diện người dùng

* Tạo các điều khiển trong Form để nhập văn bản và hiển thị ảnh Captcha:
  + **TextBox** cho phép nhập văn bản.
  + **Button** để tạo ảnh Captcha khi nhấn.
  + **PictureBox** để hiển thị ảnh Captcha.

**Bước 4:** Kết nối với DLL và hiển thị Captcha

* Khi người dùng nhấn nút, gọi phương thức GenerateCaptcha từ thư viện CaptchaLib và hiển thị kết quả lên **PictureBox**.

|  |
| --- |
| private void btnGenerateCaptcha\_Click(object sender, EventArgs e)  {  string captchaText = txtCaptchaInput.Text;  Bitmap captchaImage = CaptchaLib.CaptchaGenerator.GenerateCaptcha(captchaText);  pictureBoxCaptcha.Image = captchaImage;  } |

**3. ASP.NET Web Application (.NET Framework 2.0): Login with Captcha**

**Bước 1:** Tạo dự án ASP.NET Web Application

* Tạo một dự án **ASP.NET Web Application** trong Visual Studio với .NET Framework 2.0.
* Đặt tên cho dự án là **CaptchaLoginWebApp**.

**Bước 2**: Tạo giao diện HTML cho trang login

Chúng ta sẽ cần một trang login với các trường nhập thông tin người dùng (username, password) và Captcha khi đăng nhập sai quá 3 lần.

|  |
| --- |
| **HTML (login.aspx)**  <%@ Page Language="C#" AutoEventWireup="true" CodeBehind="login.aspx.cs" Inherits="CaptchaLoginWebApp.login" %>  <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Login with Captcha</title>  <link href="mycss.css" rel="stylesheet" />  <script src="myjs.js"></script>  </head>  <body>  <div class="login-container">  <h2>Login</h2>  <form id="loginForm" method="post" action="login.aspx">  <div class="form-group">  <label for="username">Username:</label>  <input type="text" id="username" name="username" required />  </div>  <div class="form-group">  <label for="password">Password:</label>  <input type="password" id="password" name="password" required />  </div>    <!-- Captcha will be displayed here if login attempts > 3 -->  <div class="captcha-container" id="captchaContainer" style="display: none;">  <img id="captchaImage" src="" alt="Captcha" />  <div class="form-group">  <input type="text" id="captcha" class="captcha-input" placeholder="Enter Captcha" />  </div>  </div>  <div class="form-group">  <button type="submit" class="btn-login">Login</button>  </div>  </form>  </div>  </body>  </html> |

|  |
| --- |
| **CSS**  body {  font-family: Arial, sans-serif;  background-color: #f4f4f9;  padding: 50px;  }  .login-container {  max-width: 400px;  margin: 0 auto;  background: #fff;  padding: 20px;  border-radius: 10px;  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);  }  h2 {  text-align: center;  }  .form-group {  margin-bottom: 20px;  }  .form-group input {  width: 100%;  padding: 10px;  margin: 5px 0;  border-radius: 5px;  border: 1px solid #ccc;  }  .captcha-container {  text-align: center;  }  .captcha-container img {  margin-bottom: 10px;  }  .captcha-input {  width: 100%;  padding: 10px;  margin-bottom: 20px;  border-radius: 5px;  border: 1px solid #ccc;  }  .btn-login {  width: 100%;  padding: 10px;  background-color: #5cb85c;  color: white;  border: none;  border-radius: 5px;  }  .btn-login:hover {  background-color: #4cae4c;  } |

|  |
| --- |
| **myjs.js (tương tác với API để tạo ra captcha và thực hiện đăng nhập)**  // Hàm để tải ảnh captcha từ API và hiển thị lên giao diện  function loadCaptcha() {  $.ajax({  url: 'api.aspx', // Địa chỉ API để tạo và trả về ảnh captcha  type: 'GET',  success: function (data) {  // Chuyển đổi dữ liệu trả về (Base64) thành ảnh và hiển thị lên thẻ img  var base64Image = 'data:image/png;base64,' + data;  $('#captchaImage').attr('src', base64Image); // Gán nguồn ảnh cho thẻ img  }  });  }  // Kiểm tra số lần đăng nhập thất bại và hiển thị captcha nếu cần thiết  $(document).ready(function () {  var failedAttempts = sessionStorage.getItem('failedAttempts') || 0; // Lấy số lần đăng nhập thất bại từ sessionStorage  if (failedAttempts >= 3) {  $('#captchaContainer').show(); // Hiển thị phần captcha khi số lần đăng nhập thất bại >= 3  loadCaptcha(); // Tải ảnh captcha  }  // Xử lý sự kiện khi form đăng nhập được submit  $('#loginForm').submit(function (event) {  event.preventDefault(); // Ngừng hành động mặc định của form  var username = $('#username').val(); // Lấy giá trị từ trường nhập username  var password = $('#password').val(); // Lấy giá trị từ trường nhập password  var captcha = $('#captcha').val(); // Lấy giá trị từ trường nhập captcha  // Nếu cần captcha, kiểm tra xem người dùng đã nhập chưa  if (failedAttempts >= 3 && captcha === "") {  $.alert("Vui lòng nhập captcha."); // Hiển thị cảnh báo nếu captcha chưa được nhập  return false;  }  // Gửi dữ liệu đăng nhập qua AJAX  $.ajax({  url: 'login.aspx',  method: 'POST',  data: {  username: username,  password: password,  captcha: captcha  },  success: function (response) {  if (response === 'success') {  window.location.href = 'home.aspx'; // Chuyển hướng đến trang home nếu đăng nhập thành công  } else {  failedAttempts++; // Tăng số lần thất bại khi đăng nhập sai  sessionStorage.setItem('failedAttempts', failedAttempts); // Lưu lại số lần thất bại trong sessionStorage  $.alert("Thông tin đăng nhập không đúng. Vui lòng thử lại.");  if (failedAttempts >= 3) {  $('#captchaContainer').show(); // Hiển thị captcha nếu số lần thất bại >= 3  loadCaptcha(); // Tải lại ảnh captcha  }  }  }  });  });  }); |

|  |
| --- |
| **api.aspx**  <%@ Page Language="C#" AutoEventWireup="true" CodeBehind="api.aspx.cs" Inherits="CaptchaLoginWebApp.api" %>  <%  // C# code to generate captcha image and return it as Base64  using System;  using System.Drawing;  using System.IO;  protected void Page\_Load(object sender, EventArgs e)  {  // Tạo chuỗi captcha ngẫu nhiên gồm 6 ký tự  string captchaText = GenerateRandomString(6);  Session["CaptchaText"] = captchaText; // Lưu captcha vào session để đối chiếu sau này  // Tạo ảnh chứa captcha  using (Bitmap bmp = new Bitmap(200, 50)) // Tạo một ảnh có kích thước 200x50  {  using (Graphics g = Graphics.FromImage(bmp)) // Sử dụng đối tượng Graphics để vẽ lên ảnh  {  g.Clear(Color.White); // Đặt nền trắng cho ảnh  g.DrawString(captchaText, new Font("Arial", 24), Brushes.Black, 10, 10); // Vẽ chuỗi captcha lên ảnh  using (MemoryStream ms = new MemoryStream()) // Sử dụng MemoryStream để lưu ảnh vào bộ nhớ  {  bmp.Save(ms, System.Drawing.Imaging.ImageFormat.Png); // Lưu ảnh dưới dạng PNG  byte[] imageBytes = ms.ToArray(); // Chuyển đổi ảnh thành mảng byte  string base64String = Convert.ToBase64String(imageBytes); // Chuyển mảng byte thành chuỗi Base64  Response.ContentType = "image/png"; // Đặt Content-Type là image/png  Response.BinaryWrite(imageBytes); // Gửi ảnh dưới dạng nhị phân về client  }  }  }  }  // Hàm tạo chuỗi captcha ngẫu nhiên với độ dài đầu vào  private string GenerateRandomString(int length)  {  Random random = new Random();  string characters = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789"; // Các ký tự có thể có trong captcha  char[] stringChars = new char[length];  for (int i = 0; i < length; i++)  {  stringChars[i] = characters[random.Next(characters.Length)]; // Chọn ngẫu nhiên ký tự trong chuỗi  }  return new string(stringChars); // Trả về chuỗi captcha  }  %> |

|  |  |
| --- | --- |
| **Mã QR dẫn tới github for me – bài 3** | **Link github for me** |
| A qr code on a white background  Description automatically generated | https://github.com/thangtran1904/Giamsatk57kmt |