



## A. MỤC TIÊU:

- Xây dựng Module Quản lý đăng nhập

## B. DỤNG CỤ - THIẾT BỊ THỰC HÀNH CHO MỘT SV:

STT	Chủng loại – Quy cách vật tư	Số lượng	Đơn vị	Ghi chú
1	Computer	1	1	

## C. NỘI DUNG THỰC HÀNH

### 1. Tóm tắt lý thuyết

#### 1.1. Khái niệm tầng (tier) và lớp (layer)

3-tiers là một kiến trúc kiểu client/server mà trong đó giao diện người dùng (UI-user interface), các quy tắc xử lý (BR-business rule hay BL-business logic), và việc lưu trữ dữ liệu được phát triển như những module độc lập, và hầu hết là được duy trì trên các nền tảng độc lập, và mô hình 3 tầng (3-tiers) được coi là một kiến trúc phần mềm và là một mẫu thiết kế.

Đây là kiến trúc triển khai ứng dụng ở mức vật lý. Kiến trúc gồm 3 module chính và riêng biệt:

- Tầng Presentation: hiển thị các thành phần giao diện để tương tác với người dùng như tiếp nhận thông tin, thông báo lỗi, ...
- Tầng Business Logic: thực hiện các hành động nghiệp vụ của phần mềm như tính toán, đánh giá tính hợp lệ của thông tin, ... Tầng này còn di chuyển, xử lý thông tin giữa 2 tầng trên dưới.
- Tầng Data: nơi lưu trữ và trích xuất dữ liệu từ các hệ quản trị CSDL hay các file trong hệ thống. Cho phép tầng Business logic thực hiện các truy vấn dữ liệu.

Người dùng vẫn hay nhầm lẫn giữa tier và layer vì cấu trúc phân chia giống nhau (presentation, bussiness , data). Tuy nhiên, thực tế chúng hoàn toàn khác nhau. Nếu 3 tiers có tính vật lý thì 3 layer có tính logic. Nghĩa là ta phân chia ứng dụng thành các phần (các lớp) theo chức năng hoặc vai trò một cách logic. Các layer khác nhau được thực thi trong 1 phân vùng bộ nhớ của process. Vì thế nên một tier có thể có nhiều layer.

#### 1.2. Giới thiệu mô hình 3-layer (3 lớp)

Mô hình 3-layer gồm có 3 phần chính:

- *Presentation Layer (GUI)*: Lớp này có nhiệm vụ chính giao tiếp với người dùng. Nó gồm các thành phần giao diện ( winform, webform,...) và thực hiện các công

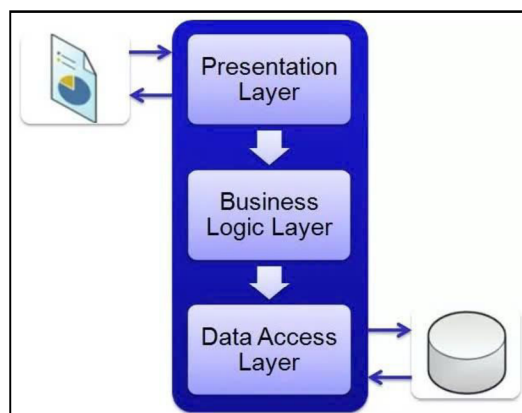
việc như nhập liệu, hiển thị dữ liệu, kiểm tra tính đúng đắn dữ liệu trước khi gọi lớp Business Logic Layer (BLL).

- *Business Logic Layer (BLL)*: Layer này phân ra 2 thành nhiệm vụ:

Đây là nơi đáp ứng các yêu cầu thao tác dữ liệu của GUI layer, xử lý chính nguồn dữ liệu từ Presentation Layer trước khi truyền xuống Data Access Layer và lưu xuống hệ quản trị CSDL.

Đây còn là nơi kiểm tra các ràng buộc, tính toàn vẹn và hợp lệ dữ liệu, thực hiện tính toán và xử lý các yêu cầu nghiệp vụ, trước khi trả kết quả về Presentation Layer.

- *Data Access Layer (DAL)*: Lớp này có chức năng giao tiếp với hệ quản trị CSDL như thực hiện các công việc liên quan đến lưu trữ và truy vấn dữ liệu ( tìm kiếm, thêm, xóa, sửa,...).



Ngoài ra, trong quá trình xây dựng, thông thường chúng ta tạo thêm một lớp có tên là DTO (Data Transfer Object).

- *DTO Layer*: Lớp này chỉ là lớp phụ, để chứa các đối tượng dùng để truyền dữ liệu giữa các lớp. Đây là lớp định nghĩa các table trong database, định nghĩa cột của nó cũng như để gán dữ liệu khi truy vấn.

### 1.3. Ưu điểm

- Việc phân chia thành từng lớp giúp cho code được tường minh hơn, từng lớp đảm nhận các chức năng khác nhau và riêng biệt như giao diện, xử lý, truy vấn nhằm giảm sự kết dính.

- Dễ bảo trì khi được phân chia, thì một thành phần của hệ thống sẽ dễ thay đổi. Việc thay đổi này có thể được cô lập trong 1 lớp, hoặc ảnh hưởng đến lớp gần nhất mà không ảnh hưởng đến cả chương trình.

- Dễ phát triển, tái sử dụng.

- Dễ bàn giao. Nếu mọi người đều theo một quy chuẩn đã được định sẵn, thì công việc bàn giao, tương tác với nhau sẽ dễ dàng hơn và tiết kiệm được nhiều thời gian.

- Dễ phân phối khối lượng công việc. Mỗi một nhóm, một bộ phận sẽ nhận một nhiệm vụ trong mô hình 3 lớp. Việc phân chia rõ ràng như thế sẽ giúp các lập trình viên kiểm soát được khối lượng công việc của mình.

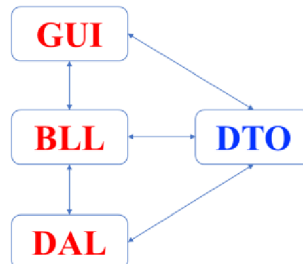
#### 1.4. Tạo một ứng dụng theo mô hình 3 lớp

Khi xây dựng ứng dụng theo mô hình 3 lớp:

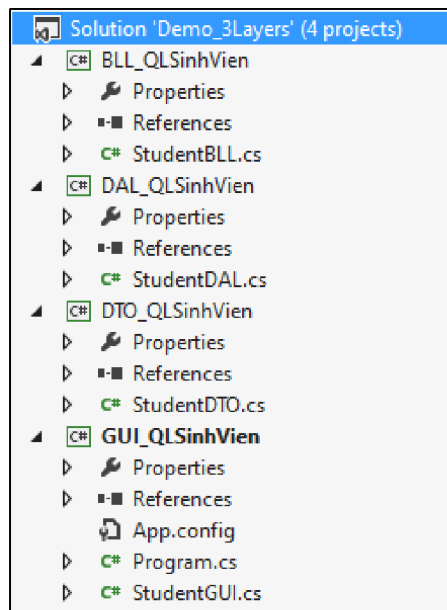
- Tạo một solution riêng (VD: Demo\_3Layers)
- 4 project khác nhau để làm nên 4 lớp, với tên các project được đặt như sau:
  - Lớp GUI (dạng **Windows Forms**): GUI\_\* (VD: GUI\_QLSinhVien)
  - Lớp Business (dạng **Class Library**): BUS\_\* (VD: BUS\_QLSinhVien)
  - Lớp Data Access (dạng **Class Library**): DAL\_\* (VD: DAL\_QLSinhVien)
  - Lớp DTO (dạng **Class Library**): DTO\_\* (VD: DTO\_QLSinhVien)
- Bên trong 4 lớp như trên, khi đặt tên cho các file \*.cs cần có các hậu tố như sau:

Ví dụ: ta có một đối tượng tên là Student thì:

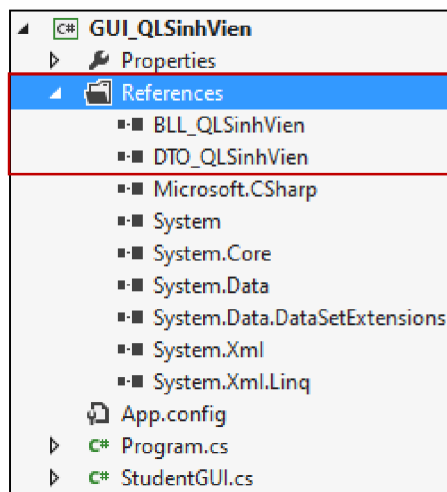
- Lớp GUI: \*GUI (VD: StudentGUI.cs)
  - Lớp Business: \*BUS (VD: StudentBLL.cs)
  - Lớp Data Access: \*DAL (VD: StudentDAL.cs)
  - Lớp DTO: \*DTO (VD: StudentDTO.cs)
- Sơ đồ liên kết trong mô hình 3 lớp hoạt động như sau:
    - GUI Layer: Liên kết tới BLL và DTO.
    - Business Layer: Liên kết tới DAL và DTO.
    - Data Access Layer: Chỉ liên kết tới DTO.



**Kết quả sau khi xây dựng:**



Cách liên kết các lớp lại với nhau:



## 2. Bài tập có hướng dẫn

Chạy File Scrip DB\_QLPQNguoiDung đính kèm, thực hiện nhập dữ liệu cho các bảng.

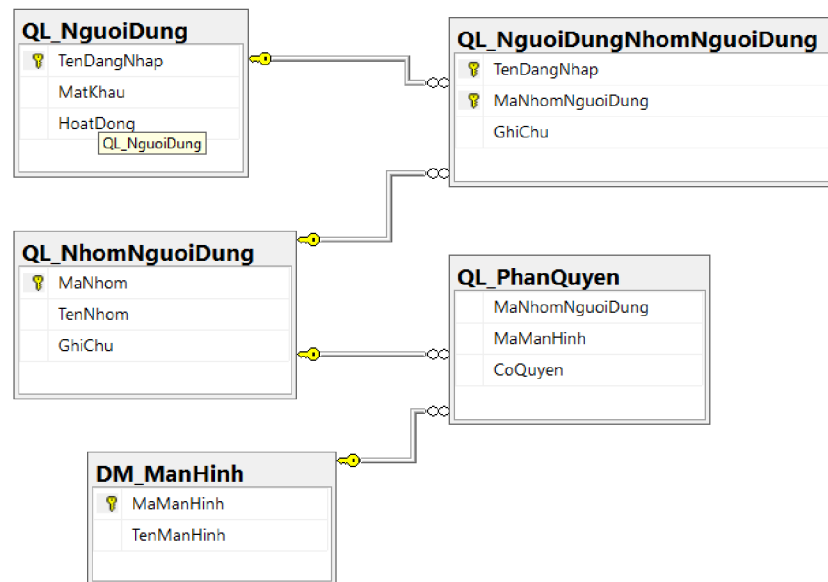
DM\_ManHinh(MaManHinh, TenManHinh)

QL\_NguoiDung (TenDangNhap, MatKhau, HoatDong)

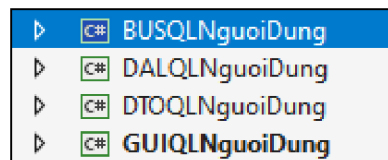
QL\_NhomNguoiDung (MaNhom, TenNhom, GhiChu)

QL\_NguoiDungNhomNguoiDung(TenDangNhap, MaNhomNguoiDung, GhiChu)

QL\_PhanQuyen (MaNhomNguoiDung, MaManHinh, CoQuyền)



**Tạo solution và các project như sau:**

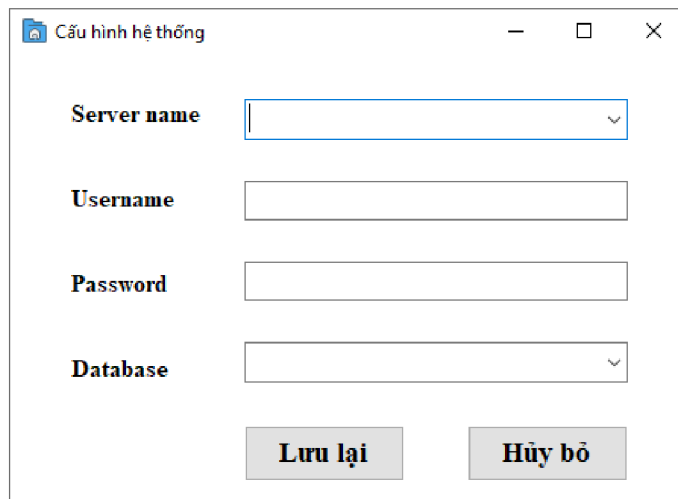


với project GUIQLNguoiDung dạng Windows Form App, các project còn lại dạng Class Library và thực hiện liên kết giữa các project theo mô hình 3 lớp.

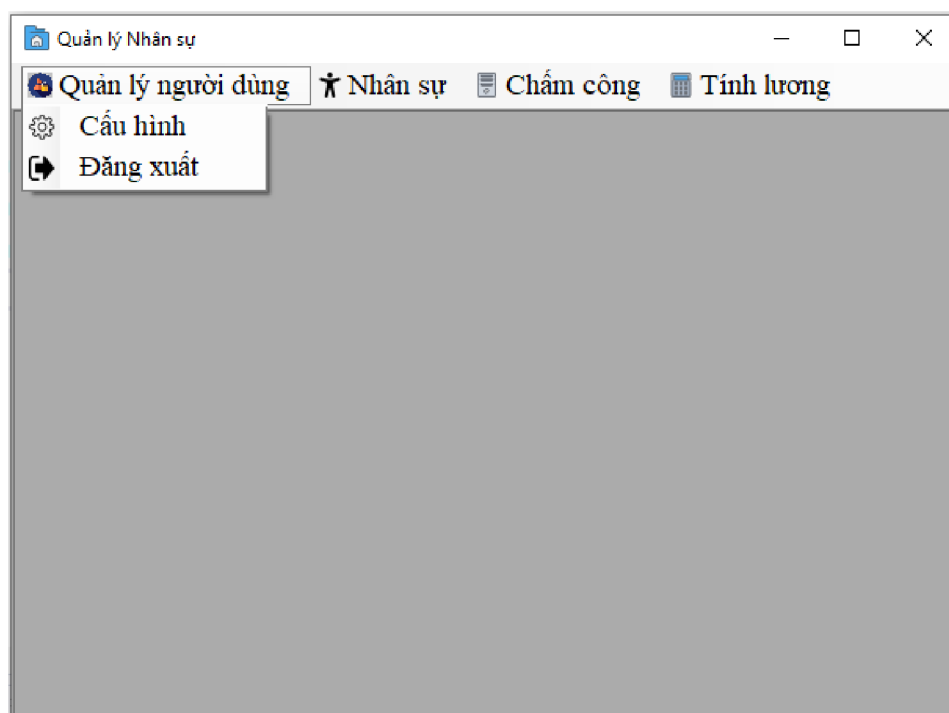
### Bước 1. Thiết kế các giao diện

Hình 1. Giao diện Đăng nhập

🔗 Lưu ý: Form đăng nhập có thể sử dụng user control login trong bài 2.



Hình 2. Giao diện Cấu hình chuỗi kết nối



Hình 3. Giao diện màn hình chính

✎ Tạo chuỗi kết nối trong tập tin “App.config”

```
<connectionStrings>
  <add name = "connString" connectionString = "Server = ...;
                                     Database = ...; User ID = ...; Pwd = ..." />
</connectionStrings>
```

Hoặc tạo biến cấu hình hệ thống trong Settings của project GUIQLNguoiDung và gán giá trị mặc định theo cấu hình kết nối SQL Server của máy tính

Name	Type	Scope	Value
STRConn	string	User	Server=TenServer;Database=DB_QLPQNguoiDung;User ID=sa; Pwd=1

**Quá trình xử lý thông tin đăng nhập**

1. Kiểm tra thông tin đăng nhập. (Nếu textbox Tên đăng nhập hoặc Mật khẩu bỏ trống thì yêu cầu nhập lại).
2. Kiểm tra thông tin chuỗi kết nối
  - 0: Kết nối thành công, chuỗi cấu hình hợp lệ.
  - 1: Chuỗi cấu hình không tồn tại.
  - 2: Chuỗi cấu hình không phù hợp.
3. Cấu hình chuỗi kết nối phù hợp: Xử lý đăng nhập trường hợp 0.
  - Kiểm tra sự tồn tại của Tên người dùng và mật khẩu
  - Tên người dùng còn hoạt động hay không?
4. Cấu hình chuỗi kết nối không phù hợp: Xử lý tạo mới cấu hình nếu là trường hợp 1, 2

**Bước 2.** Tạo class `NguoIDungDTO` trong project `DTOQLNguoIDung` gồm 3 thuộc tính lưu trữ tên đăng nhập, mật khẩu và tình trạng hoạt động của đối tượng người dùng hệ thống.

```
public class NguoIDungDTO
{
    public string User { get; set; }
    public string Pass { get; set; }
    public NguoIDungDTO()
    {
        User = string.Empty;
        Pass = string.Empty;
    }
    public NguoIDungDTO(string user, string pass)
    {
        User = user;
        Pass = pass;
    }
}
```

**Bước 3.** Thực hiện sự kiện cho hành động đăng nhập

```
if (string.IsNullOrEmpty(txtUsername.Text.Trim()))
{
    MessageBox.Show("Không được bỏ trống " + lblUsername.Text.ToLower());
    this.txtUsername.Focus();
    return;
}
if (string.IsNullOrEmpty(this.txtPassword.Text))
{
    MessageBox.Show("Không được bỏ trống " + lblPassword.Text.ToLower());
}
```

```

        this.txtPassword.Focus();
        return;
    }
    string strConn = Properties.Settings.Default.STRConn;
    int kq = busNguoiDung.checkConfig(strConn); /* busNguoiDung và
    CheckConfig(string) là đối tượng và phương thức của lớp BUSNguoiDung trong
    project BUSQLNguoiDung */
    if (kq == 0)
        ProcessLogin(); //Cấu hình phù hợp xử lý đăng nhập
    else
    {
        if (kq == 2)
            MessageBox.Show("Chuỗi cấu hình không phù hợp");
        else
            MessageBox.Show("Chưa cấu hình hệ thống");
        ProcessConfig(); //Xử lý cấu hình: gọi form Cấu hình
    }
}

```

#### Lớp BUSNguoiDung

```

public class BUSNguoiDung
{
    NguoiDungDAL daNguoiDung = new NguoiDungDAL();
    public int checkConfig(string strConn)
    {
        return daNguoiDung.Check_Config(strConn); /* daNguoiDung và
        Check_Config(string) là đối tượng và phương thức của lớp
        NguoiDungDAL trong project DALQLNguoiDung */
    }
}

```

#### Lớp NguoiDungDAL

```

public class NguoiDungDAL
{
    SqlConnection _Sqlconn;
    public NguoiDungDAL()
    {
        _Sqlconn = new SqlConnection();
    }
    public int Check_Config(String strConn)
    {
        if (strConn == string.Empty)

```



```

        return 1; //Chuỗi cấu hình không tồn tại
        _Sqlconn = new SqlConnection(strConn);
        try
        {
            if (_Sqlconn.State == ConnectionState.Closed)
                _Sqlconn.Open();
            return 0; //Kết nối thành công chuỗi cấu hình hợp lệ
        }
        catch
        {
            return 2; //Chuỗi cấu hình không phù hợp
        }
    }
}

```

**Bước 4.** Cấu hình chuỗi kết nối phù hợp Xử lý đăng nhập trường hợp 0:

Tạo project UtilityTools dạng Class Library chứa enum `LoginResult`

```

public enum LoginResult
{
    /// Wrong username or password
    Invalid,
    /// User had been disabled
    Disabled,
    /// Logging success
    Success
}

```

### Bước 5. Viết hàm ProcessLogin()

```
void ProcessLogin()
{
    LoginResult result;
    NguoIDungDTO nguoiDung = new NguoIDungDTO(txtUsername.Text, txtPassword.Text);
    result = busNguoiDung.checkLogin(nguoiDung);
    //Wrong username or pass
    if (result == LoginResult.Invalid)
    {
        MessageBox.Show("Tài khoản bị khóa");
        return;
    }
    //Account had been disabled
    else if (result == LoginResult.Disabled)
    {
        MessageBox.Show("Sai thông tin đăng nhập");
        return;
    }
    if (Program.mainForm == null || Program.mainForm.IsDisposed)
    {
        Program.mainForm = new frmMain();
    }
    this.Visible = false;
    Program.mainForm.Show();
}
```

### Bước 6. Hàm checkLogin() trong class BUSNguoiDung

```
public LoginResult checkLogin(NguoiDungDTO nguoiDung)
{
    return daNguoiDung.Check_User(nguoiDung.User, nguoiDung.Pass);
}
```

### Bước 7. Hàm Check\_User() trong NguoIDungDAL

```
public LoginResult Check_User(string pUser, string pPass)
{
    string _sqlstring = "select * from QL_NguoiDung where TenDangNhap = '" + pUser + "' and MatKhai = '" + pPass + "'";
    SqlDataAdapter daUser = new SqlDataAdapter(_sqlstring, _Sqlconn);

    DataTable dt = new DataTable();
    daUser.Fill(dt);
}
```

```

if (dt.Rows.Count == 0)
    return UtilityTools.LoginResult.Disabled; // User không tồn tại
else if (dt.Rows[0][2] == null || dt.Rows[0][2].ToString() == "0")
{
    return UtilityTools.LoginResult.Invalid; //Không hoạt động
}
return UtilityTools.LoginResult.Success; // Đăng nhập thành công
}

```

**Bước 8.** Hàm ProcessConfig()

```

private void ProcessConfig()
{
    frmCauHinh frmCau = new frmCauHinh();
    frmCau.Show();
}

```

**Bước 9.** Class Program được điều chỉnh như sau để thực thi form Đăng nhập:

```

static class Program
{
    public static frmMain mainForm = null;
    public static LoginForm loginForm = null;
    /// <summary>
    /// The main entry point for the application.
    /// </summary>
    [STAThread]
    static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        loginForm = new LoginForm();
        Application.Run(loginForm);
    }
}

```

**Bước 10.** Cấu hình chuỗi kết nối không phù hợp, xử lý tạo mới cấu hình nếu là trường hợp 1 và 2, mở form Cấu hình:

Server name	<input type="text"/>
Username	<input type="text"/>
Password	<input type="text"/>
Database	<input type="text"/>
<input type="button" value="Lưu lại"/> <input type="button" value="Hủy bỏ"/>	

Quá trình thực trên Form Cấu hình:

1. Tìm kiếm Server Name
2. Nhập User name
3. Nhập Password tương ứng bước 2
4. Tìm kiếm Database tương ứng với các thông tin hợp lệ trên
5. Tất cả hợp lệ lưu lại chuỗi cấu hình.

1. Tìm kiếm Server Name

```
private void cboServer_DropDown(object sender, EventArgs e)
{
    cboServer.DataSource = CauHinh.GetServerName();
    cboServer.DisplayMember = "ServerName";
}
```

```
public DataTable GetServerName()
{
    DataTable dt = new DataTable();
    dt = SqlDataSourceEnumerator.Instance.GetDataSources();
    return dt;
}
```

🔔 **Lưu ý:** bật services SQL Server Browser.

2. Nhập User name
3. Nhập Password tương ứng bước 2
4. Tìm kiếm Database tương ứng với các thông tin hợp lệ trên

```
private void cboDataBase_DropDown(object sender, EventArgs e)
{
    cboDataBase.DataSource = GetDBName(cboServer.Text,
                                       txtUsername.Text, txtPassword.Text);
    cboDataBase.DisplayMember = "name";
}
```

```

public DataTable GetDBName(string pServer, string pUser, string pPass)
{
    DataTable dt = new DataTable();
    SqlDataAdapter da = new SqlDataAdapter("select name from
        sys.Databases", "Data Source = " + pServer + "; Initial Catalog =
        master; User ID=" + pUser + "; pwd = " + pPass);
    da.Fill(dt);
    return dt;
}

```

##### 5. Tất cả hợp lệ lưu lại chuỗi cấu hình

```

private void btnLuu_Click(object sender, EventArgs e)
{
    // Có thể kiểm tra chuỗi cấu hình hợp lệ thì mới lưu
    SaveConfig(cboServer.Text, txtUsername.Text, txtPassword.Text,
        cboDataBase.Text);
    this.Close();
}

```

```

public void SaveConfig(string pServer, string pUser, string pPass,
    string pDBname)
{
    Properties.Settings.Default.STConn = "Data Source=" +
        pServer + "; Initial Catalog=" + pDBname + ";
        User ID=" + pUser + "; pwd = " + pPass + "";
    Properties.Settings.Default.Save();
}

```

### Bổ sung: Mã hóa Password dùng MD5

Tạo class MD5Hash trong UtilityTools

```

public class MD5Hash
{
    public static string Hash(string text)
    {
        MD5 md5 = MD5.Create();
        byte[] hash = md5.ComputeHash(Encoding.UTF8.GetBytes(text));
        StringBuilder hashSb = new StringBuilder();
        foreach (byte b in hash)
        {

```

```
        hashSb.Append(b.ToString("X2"));
    }
    return hashSb.ToString();
}
}
```