**TRƯỜNG ĐẠI HỌC HÀNG HẢI VIỆT NAM  
KHOA CÔNG NGHỆ THÔNG TIN**

**-----\*\*\*\*\*-----**

****

**BÁO CÁO BÀI TẬP LỚN  
HỌC PHẦN “LẬP TRÌNH WINDOW”**

***Đề tài:***

***XÂY DỰNG ỨNG DỤNG LẬP LỊCH CÔNG VIỆC CÁ NHÂN***

***GVHD: ThS. Nguyễn Kim Anh  
Sinh viên thực hiện : Hoàng Tiến Dũng - Mã sv: 63628***

***Nguyễn Viết Phương - Mã sv: 63615***

***Hải Phòng, tháng 04 năm 2018***

**TRƯỜNG ĐẠI HỌC HÀNG HẢI VIỆT NAM**

**KHOA CÔNG NGHỆ THÔNG TIN  
BỘ MÔN KHOA HỌC MÁY TÍNH**

**-----\*\*\*\*\*-----**

**BÀI TẬP LỚN**

**HỌC PHẦN: LẬP TRÌNH WINDOW**

1. **Tên đề tài**

***Xây dựng ứng dụng lập lịch công việc cá nhân***

1. **Mục đích**

Tìm hiểu và xây dựng một ứng dụng lập lịch công việc hàng ngày để thông báo những việc cần làm trong ngày sử dụng ngôn ngữ lập trình C# Winform.

1. **Công việc cần thực hiện**

* Tìm hiểu về lập trình Winform, Dlegate-Event, Timer, NotifyIcon, xử lí ngày tháng năm,…
* Thiếu kế giao diện chính cho Calendar.
* Thiết kế form Plan để hiển thị lịch công việc trong ngày.
* Tạo UseControl để xử lý một công việc.
* Tạo ma trận hiển thị lịch theo ngày được chọn, tạo cấu trúc công việc, lưu trữ công việc, thêm sửa xóa công việc, tạo thông báo.
* Tạo class PlanItem để lưu trữ từng công việc.
* Tạo class PlanData để chưa danh sách các công việc.
* Phân chia công việc của nhóm và thực hiện đúng tiến độ đề ra.
* Làm báo cáo bài tập lớn.
* Bảo vệ bài tập lớn.

1. **Yêu cầu**

* Kết quả làm bài tập lớn: Báo cáo bài tập lớn
* Báo cáo bài tập lớn phải được trình bày theo mẫu quy định (kèm theo), báo cáo có thể xuất thành tệp định dạng PDF và nộp qua email (không bắt buộc phải in ấn)
* Hạn nộp báo cáo: Tuần thứ 14

**5. Tài liệu tham khảo**

* Bài giảng “**Lập trình Window C#.NET**”, khoa CNTT, ĐHHHVN
* Tìm hiểu trên Internet.

***Hải phòng, tháng 04 năm 2018***

**NGƯỜI HƯỚNG DẪN**

**MỤC LỤC**

[DANH MỤC CÁC HÌNH VẼ i](#_Toc514190278)

[GIỚI THIỆU 0](#_Toc514190279)

[CHƯƠNG 1. CƠ SỞ LÍ THUYẾT 1](#_Toc514190280)

[**1.1** **Tổng quan về ngôn ngữ lập trình C#** 1](#_Toc514190281)

[**1.1.1.** **Những đặc trưng cơ bản của ngôn ngữ lập trình C#** 1](#_Toc514190282)

[**1.1.2.** **Ứng dụng chính của C#** 2](#_Toc514190283)

[**1.2** **Giới thiệu về Windows Form của kiến trúc .NET** 3](#_Toc514190284)

[CHƯƠNG 2: PHÂN TÍCH BÀI TOÁN 4](#_Toc514190285)

[**2.1** **Tổng quan về chương trình** 4](#_Toc514190286)

[**2.2** **Phân tích các chức năng cơ bản** 4](#_Toc514190287)

[CHƯƠNG 3: THIẾT KẾ GIAO DIỆN VÀ CÀI ĐẶT CHƯƠNG TRÌNH 5](#_Toc514190288)

[**3.1** **Thiết kế chương trình** 5](#_Toc514190289)

[**3.1.1** **Giao diện chính của chương trình** 5](#_Toc514190290)

[**3.1.2** **Giao diện lịch công việc** 6](#_Toc514190291)

[**3.1.3** **Giao diện UseControl công việc** 6](#_Toc514190292)

[**3.1.4** **Tính năng thông báo** 7](#_Toc514190293)

[**3.1.5** **Tính năng khởi động cùng Windows** 8](#_Toc514190294)

[**3.2** **Cài đặt chương trình** 9](#_Toc514190295)

[KẾT LUẬN 27](#_Toc514190296)

[TÀI LIỆU THAM KHẢO 28](#_Toc514190297)

**DANH MỤC CÁC HÌNH VẼ**

[Hình 1: Giao diện chính chương trình 5](#_Toc514190052)

[Hình 2: Lịch công việc 6](#_Toc514190053)

[Hình 3: UseControl xử lý công việc 6](#_Toc514190054)

[Hình 4: Thông báo công việc 7](#_Toc514190055)

[Hình 5: Khởi động cùng hệ thống 8](#_Toc514190056)

**GIỚI THIỆU**

***Bài toán :***

Ứng dụng lập lịch công việc được xây dựng bằng ngôn ngữ C# Winform.

Để xây dựng ứng dụng lập lịch công việc ta cần xây dựng các lớp với các thuộc tính tương ứng :

* **PlanItem** : Row, Column, sizeBtnWidth, sizeBtnHeight, margin, notifyTime, timeOut.
* **PlanData** : List<PlanItiem> ListJob.

***Chức năng :***

Xây dựng lớp LinkList để lưu trữ dữ liệu là các số nguyên với các phương thức :

* Xem lịch hàng ngày.
* Ghi chú công việc hàng ngày.
* Thêm, sửa, xóa công việc.
* Lưu trữ công việc.
* Thông báo những công việc cần làm trong ngày.
* Khởi động cùng hệ thống.

***Nội dung nghiên cứu:***

* Kiến thức về lập trình C# Winform cơ bản, Dlegate-Event, Timer, NotifyIcon, xử lí ngày tháng năm,…

***Phương pháp nghiên cứu:***

* Bài giảng “**Lập trình Window C#.NET**”, khoa CNTT, ĐHHHVN.
* Tìm kiếm tài liệu và nghiên cứu trên mạng Internet.

**CHƯƠNG 1. CƠ SỞ LÍ THUYẾT**

* 1. **Tổng quan về ngôn ngữ lập trình C#**
* “Ngôn ngữ C# là một ngôn ngữ lập trình hiện đại, đơn giản, mục đích tổng quát, hướng đối tượng được phát triển bởi Microsoft, là phần khởi đầu cho kế hoạch .NET của họ và được phê chuẩn bởi European Computer Manufacturers Association (ECMA) và International Standards Organization (ISO)
* Ngôn ngữ C# ra đời nằm 2000, được thiết kế chủ yếu bởi [Anders Hejlsberg](https://vi.wikipedia.org/wiki/Anders_Hejlsberg) kiến trúc sư phần mềm nổi tiếng với các sản phẩm [Turbo Pascal](https://vi.wikipedia.org/wiki/Turbo_Pascal), [Delphi](https://vi.wikipedia.org/wiki/Delphi_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh)), J++, WFC.
* C# được thiết kế cho Common Language Infrastructure (CLI), mà gồm Executable Code và Runtime Environment, cho phép chúng ta sử dụng các ngôn ngữ high-level đa dạng trên các nền tảng và cấu trúc máy tính khác nhau.
* C# với sự hỗ trợ mạnh mẽ của .NET Framework giúp cho việc tạo một ứng dụng Windows Forms hay WPF (Windows Presentation Foundation), . . . trở nên rất dễ dàng.

1. **Những đặc trưng cơ bản của ngôn ngữ lập trình C#**

* ***C# là ngôn ngữ lập trình đơn giản***

Ngôn ngữ C# dựng trên nền tảng C++ và Java nên ngôn ngữ C# khá đơn giản. Nếu chúng ta thân thiện với C và C++ hoặc thậm chí là Java, chúng ta sẽ thấy C# khá giống về diện mạo, cú pháp, biểu thức, toán tử và những chức năng khác được lấy trực tiếp từ ngôn ngữ C và C++, nhưng nó đã được cải tiến để làm cho ngôn ngữ đơn giản hơn. Một vài trong các sự cải tiến là loại bỏ các dư thừa, hay là thêm vào những cú pháp thay đổi.

* ***C# là ngôn ngữ hiện đại***

Một vài khái niệm khá mới mẻ khá mơ hồ với các bạn vừa mới học lập trình, như xử lý ngoại lệ, những kiểu dữ liệu mở rộng, bảo mật mã nguồn..v..v... Đây là những đặc tính được cho là của một ngôn ngữ hiện đại cần có. Và C# chứa tất cả các đặt tính ta vừa nêu trên.

* ***C# là một ngôn ngữ lập trình thuần hướng đối tượng***

Lập trình hướng đối tượng(Object-oriented programming- OOP) là một phương pháp lập trình có 4 tính chất. Đó là tính trừu tượng (*abstraction*), tính đóng gói (encapsulation), tính đa hình (polymorphism) và tính kế thừa (inheritance). C# hỗ trợ cho chúng ta tất cả những đặc tính trên.

* ***C# là một ngôn ngữ ít từ khóa***

Là ngôn ngữ khá đơn giản, chỉ có khoảng 80 từ khóa và hơn mười mấy kiểu dữ liệu được dựng sẵn.

1. **Ứng dụng chính của C#**

* ***Console***
* Giao tiếp với người dùng bằng bàn phím.
* Không có giao diện đồ họa.

Ứng dụng Console là ứng dụng có giao diện text, chỉ xử lý nhập xuất trên màn hình Console, tương tự với các ứng dụng DOS trước đây

* ***WindowForm***
* Giao tiếp với người dùng bằng bàn phím và chuột.
* Có giao diện đồ họa và xử lý sự kiện.

Là ứng dụng được hiển thị với giao diện cửa sổ đồ họa. Chúng ta chỉ cần kéo và thả các điều khiển (control) lên cửa sổ Form. Visual Studio sẽ sinh mã trong chương trình để tạo ra, hiển thị các thành phần trên cửa sổ.”

* ***Web***
* “Kết hợp với [ASP .NET,](https://www.oktot.com/category/lap-trinh/asp-mvc/) C# đóng vài trò xử lý bên dưới (underlying code).
* Có giao diện đồ họa và xử lý sự kiện.

Môi trường .NET cung cấp công nghệ ASP.NET giúp xây dựng những trang Web động. Để tạo ra một trang ASP.NET, người lập trình sử dụng ngôn ngữ biên dịch như C# hoặc C# để viết mã. Để đơn giản hóa quá trình xây dựng giao diện người dùng cho trang Web, .NET giới thiệu công nghệ Webform. Cách thức tạo ra các Web control  tương tự như khi ta xây dựng ứng dụng trên Windows Forms.”

* 1. **Giới thiệu về Windows Form của kiến trúc .NET**

“Windows Forms là sự thay thế của .NET cho MFC trong Visual C++. Không như thư viện MFC dùng đóng gói cho tập các hàm Win32 API. Window Forms hoàn toàn là các lớp hướng đối tượng, mang tính kế thừa dành cho các nhà phát triển ứng dụng trong môi trường .NET.

Mặc dù mang thuật ngữ “Forms” nhưng việc thiết kế các thành phần trên Forms lại không dựa vào các file tài nguyên resource như khi thiết kế các biểu mẫu Dialog trong MFC truyền thống của Windows. Mỗi thành phần giao diện còn gọi là component đặt trên Form của Windows Forms là một thể hiện (instance) cụ thể của một lớp nào đó. Component được định vị trên Form và thay đổi giao diện thông qua các phương thức (method), thuộc tính (property) của lớp đối tượng. Các công cụ phát triển trực quan sẽ cho phép kéo thả (drag-drop) những thành phần component lên Form và quản lý chặt chẽ việc khởi tạo mã nguồn cũng như tương tác giữa các thành phần component này với nhau.”

**CHƯƠNG 2: PHÂN TÍCH BÀI TOÁN**

1. **Tổng quan về chương trình**

Ứng dụng lập lịch công việc nhằm hỗ trợ trong việc ghi chú những công việc hàng ngày và thông báo cho người dùng biết công việc của mình phải làm trong ngày để có thể hoàn thành công việc một cách tốt nhất hiệu quả, ứng dụng phải đc thiết kế đơn giản, thân thiện với người dùng, dễ dàng sử dụng. Gồm có những chức năng cơ bản xem lịch hàng ngày, tạo công việc trong ngày, thêm công việc,sửa công việc, xóa công việc, thông báo công việc,lưu trữ công việc, thống kê những việc cần làm trong ngày, những việc đã bỏ lỡ và đã hoàn thành…

1. **Phân tích các chức năng cơ bản**
2. **Tạo công việc**

Nhập các thông tin : nội dung công việc, thời gian, tình trạng công việc (Doing, Coming, Missed, Done)

1. **Chức năng thêm công việc**

Thêm công việc vào lịch công việc trong ngày.

1. **Chức năng sửa và xóa công việc**

Cho phép sửa thông tin nội dung công việc, thời gian công việc và tình trạng rồi cập nhật lại vào lịch công việc của ngày hôm đó hoặc xóa bỏ công việc khỏi lịch công việc.

1. **Thông báo công việc**

Chức năng này giúp nhắc nhở cho người dùng biết những công việc cần phải làm, khi đến giờ sẽ hiện lên thông báo cho người dùng biết công việc đang cần phải làm.

1. **Lưu trữ công việc**

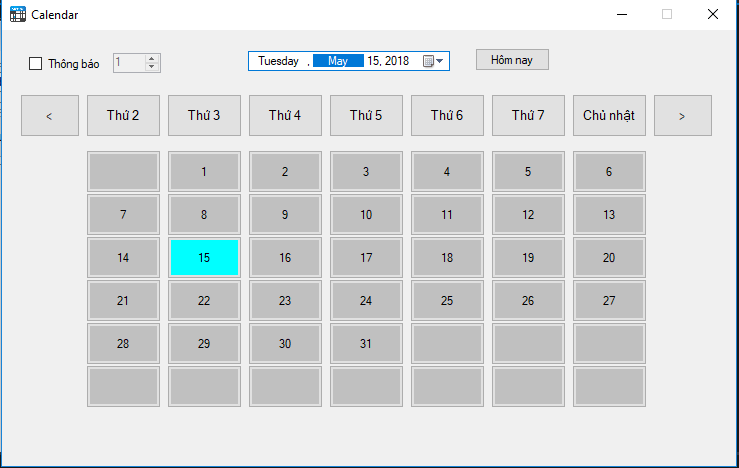
Có khả năng lưu trữ công việc của mỗi ngày người dùng có thể xem lại chúng bất cứ khi nào.

1. **Thống kê công việc**

Thông kê xem ngày hôm nay bạn có bao nhiêu việc cần làm đã hoàn thành bao nhiêu việc, bao nhiêu việc đang làm, bao nhiêu việc bỏ lỡ và bao nhiêu việc sắp đến.

**CHƯƠNG 3: THIẾT KẾ GIAO DIỆN VÀ CÀI ĐẶT CHƯƠNG TRÌNH**

1. **Thiết kế chương trình**
2. **Giao diện chính của chương trình**



Hình 1: Giao diện chính chương trình

Giao diện chính có tên là Calendar gồm:

* Checkbox thông báo có chức năng để hẹn thời thời gian thông báo
* Add ma trận các button để hiển thị lịch hàng ngày và tạo công việc cho từng ngày.

1. **Giao diện lịch công việc**



Hình 2: Lịch công việc

* Chức năng thêm việc để thêm công việc trong ngày.
* Chức năng hôm nay để trở về lịch công việc của ngày hôm nay
* Chức năng hôm qua và ngày mai để xem lịch công việc của ngày hôm qua và xem lịch công việc của ngày mai.

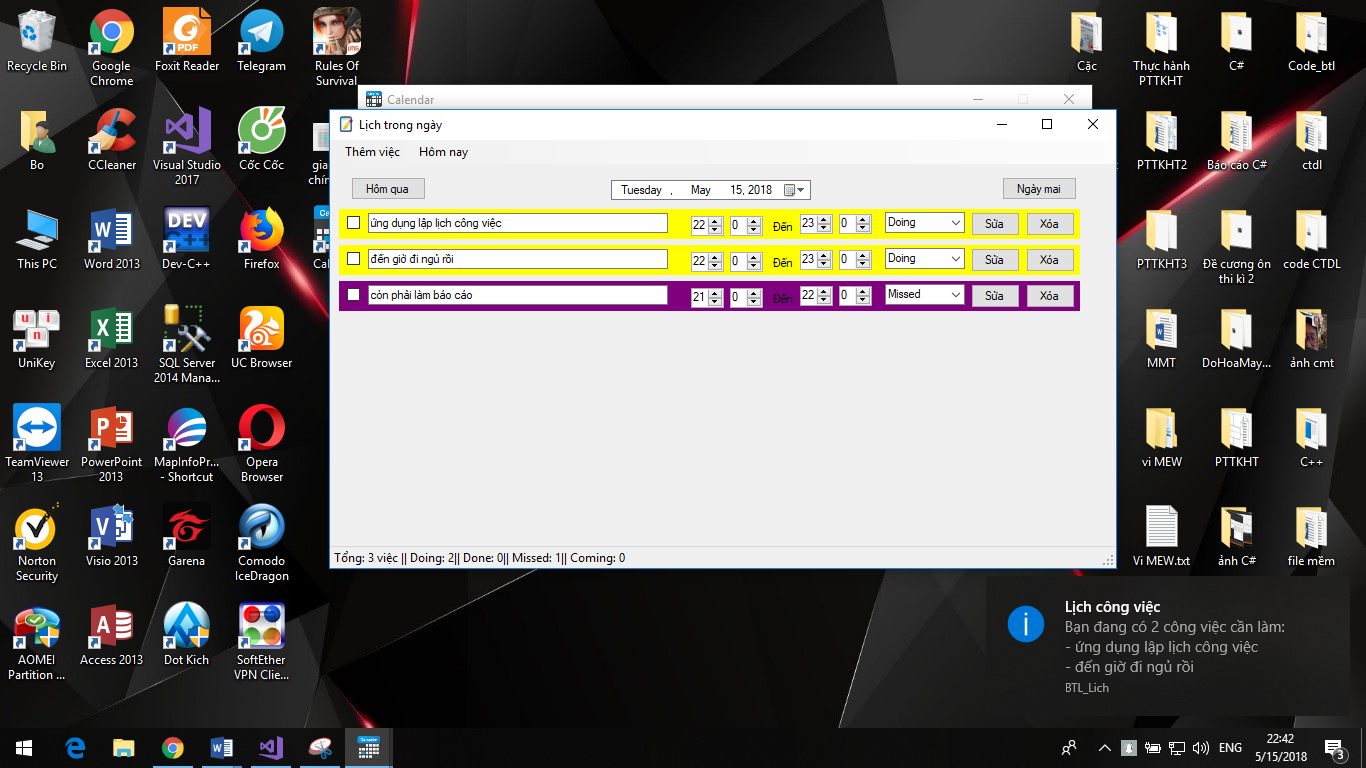
1. **Giao diện UseControl công việc**



Hình 3: UseControl xử lý công việc

Nhập nội dung công việc và các chức năng sửa xóa xử lý từng công việc.

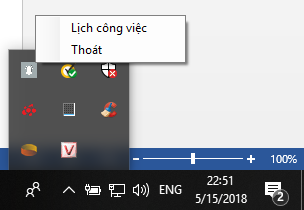
1. **Tính năng thông báo**



Hình 4: Thông báo công việc

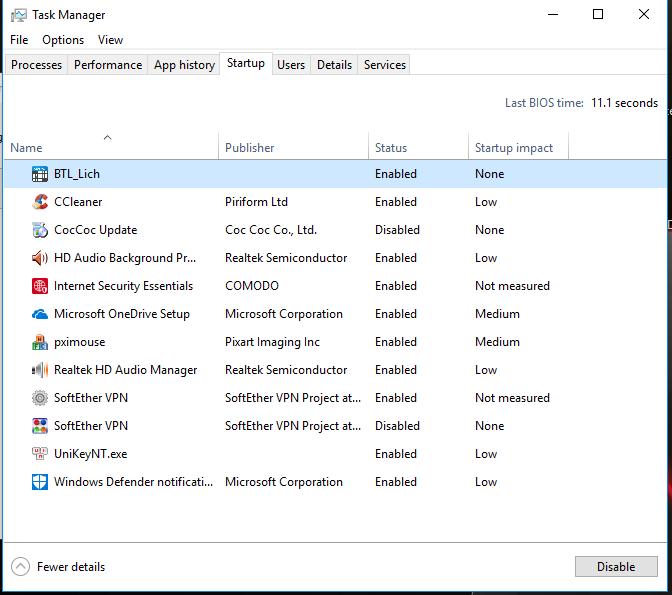
Khi tích vào thông báo trên ứng dụng thì sẽ có thông báo công việc cần làm. Nếu công việc đã quá giờ thì sẽ chuyển thành Missed (bỏ lỡ) và đổi màu.

Có thêm 1 cái NotifyIcon khi kích vào có thể xem lịch hoặc thoát chương trình:



1. **Tính năng khởi động cùng Windows**

Khi bật máy chương trình sẽ khởi động cùng hệ thống



Hình 5: Khởi động cùng hệ thống

***Đăng ký phần mềm khởi động cùng hệ thống:***

RegistryKey regkey = Registry.CurrentUser.CreateSubKey("Software\\Calendar");

RegistryKey regstart = Registry.CurrentUser.CreateSubKey("SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run");

string keyvalue = "1";

//string subkey = "Software\\ManhQuyen";

try

{

regkey.SetValue("Index", keyvalue);

regstart.SetValue("Calendar", Application.StartupPath + "\\BTL\_Lich.exe");

}

catch (System.Exception ex)

{

}

1. **Cài đặt chương trình**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

using System.Xml.Serialization;

using System.Xml;

using System.Threading;

using Microsoft.Win32;

namespace BTL\_Lich

{

public partial class Calender : Form

{

#region properties

private List<List<Button>> matrix;

private List<string> dateOfWeek = new List<string>() { "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday" };

public List<List<Button>> Matrix { get => matrix; set => matrix = value; }

private PlanData job;

private PlanItem ajob;

private int time;

public PlanData Job { get => job; set => job = value; }

public int Time { get => time; set => time = value; }

public PlanItem Ajob { get => ajob; set => ajob = value; }

private string filepath = "luutru.xml";

#endregion

public Calender()

{

InitializeComponent();

RegistryKey regkey = Registry.CurrentUser.CreateSubKey("Software\\Calendar");

//mo registry khoi dong cung win

RegistryKey regstart = Registry.CurrentUser.CreateSubKey("SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run");

string keyvalue = "1";

//string subkey = "Software\\ManhQuyen";

try

{

//chen gia tri key

regkey.SetValue("Index", keyvalue);

//regstart.SetValue("taoregistrytronghethong", "E:\\Studing\\Bai Tap\\CSharp\\Channel 4\\bai temp\\tao registry trong he thong\\tao registry trong he thong\\bin\\Debug\\tao registry trong he thong.exe");

regstart.SetValue("Calendar", Application.StartupPath + "\\BTL\_Lich.exe");

////dong tien trinh ghi key

//regkey.Close();

}

catch (System.Exception ex)

{

}

tmNotify.Start();

Time = 0;

LoadMtrix();

try

{

Job= DeserializeFromXML(filepath) as PlanData;

}

catch {

SetDefualData();

}

}

void SetDefualData()

{

/\*Job = new PlanData();

Job.ListJob = new List<PlanItem>();

Job.ListJob.Add(new PlanItem() {

Date = DateTime.Now,

FromTime=new Point(4,0),

ToTime=new Point(5,0),

Job = "test",

Status = PlanItem.list[(int)ePlanItem.Coming]

});\*/

}

void LoadMtrix()

{

Matrix = new List<List<Button>>();

Button preBtn = new Button() { Width = 0, Height = 0, Location = new Point(-Cons.margin, 0) };

for (int i = 0; i < Cons.Row; i++)

{

Matrix.Add(new List<Button>());

for (int j = 0; j < Cons.Column; j++)

{

Button button = new Button() { Width = Cons.sizeBtnWidth, Height = Cons.sizeBtnHeight };

button.Location = new Point(preBtn.Location.X + preBtn.Width + Cons.margin, preBtn.Location.Y);

pnlMaTrixNgay.Controls.Add(button);

button.Click += Button\_Click;

Matrix[i].Add(button);

preBtn = button;

}

preBtn = new Button()

{

Width = 0,

Height = 0,

Location = new Point(-Cons.margin, preBtn.Location.Y + Cons.sizeBtnHeight)

};

}

DefualtDate();

// AddNunmberMatrixByDate(dtpkDate.Value);

}

private void Button\_Click(object sender, EventArgs e)

{

if (string.IsNullOrEmpty((sender as Button).Text))

return;

Plan daily = new Plan(new DateTime(dtpkDate.Value.Year, dtpkDate.Value.Month, Convert.ToInt32((sender as Button).Text)), Job);

daily.ShowDialog();

}

int DayOfMonth(DateTime date)

{

switch (date.Month)

{

case 1:

case 3:

case 5:

case 7:

case 8:

case 10:

case 12:

return 31;

case 2:

if ((date.Year % 4 == 0 && date.Year % 100 != 0) || date.Year % 400 == 0)

return 29;

else

return 28;

default:

return 30;

}

}

void DefualtDate()

{

dtpkDate.Value = DateTime.Now;

}

bool cmpDate(DateTime a, DateTime b)

{

return a.Year == b.Year && a.Month == b.Month && a.Day == b.Day;

}

void AddNunmberMatrixByDate(DateTime date)

{

ClearMatrix();

DateTime useDate = new DateTime(date.Year, date.Month, 1);

int line = 0;

for (int i = 1; i <= DayOfMonth(date); i++)

{

int column = dateOfWeek.IndexOf(useDate.DayOfWeek.ToString());

Button button = Matrix[line][column];

button.Text = i.ToString();

if (cmpDate(useDate, DateTime.Now))

{

button.BackColor = Color.DeepPink;

}

if(cmpDate(useDate,date))

{

button.BackColor = Color.Aqua;

}

if (column >= 6)

line++;

useDate = useDate.AddDays(1);

}

}

void ClearMatrix()

{

for (int i = 0; i < Matrix.Count; i++)

{

for (int j = 0; j < Matrix[i].Count; j++)

{

Button btn = Matrix[i][j];

btn.Text = "";

btn.BackColor = Color.Silver;

//btn.BackgroundImage = Image.FromFile(@"E:\explosion\_5-wallpaper-1920x1080.jpg");

// btn.BackgroundImageLayout = ImageLayout.Stretch;

}

}

}

private void dtpkDate\_ValueChanged(object sender, EventArgs e)

{

AddNunmberMatrixByDate((sender as DateTimePicker).Value);

}

private void cmdToDay\_Click(object sender, EventArgs e)

{

dtpkDate.Value = DateTime.Now;

}

private void SerializeToXML(object data,string filepath)

{

FileStream file = new FileStream(filepath, FileMode.Create, FileAccess.Write);

XmlSerializer xml = new XmlSerializer(typeof(PlanData));

xml.Serialize(file, data);

file.Close();

}

private object DeserializeFromXML(string filepath)

{

FileStream file = new FileStream(filepath, FileMode.Open, FileAccess.Read);

try

{

XmlSerializer xml = new XmlSerializer(typeof(PlanData));

object result = xml.Deserialize(file);

file.Close();

return result;

}

catch

{

file.Close();

throw new NotImplementedException();

}

}

private void cmpNextMonth\_Click(object sender, EventArgs e)

{

dtpkDate.Value = dtpkDate.Value.AddMonths(1);

}

private void cmpPreMonth\_Click(object sender, EventArgs e)

{

dtpkDate.Value = dtpkDate.Value.AddMonths(-1);

}

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

SerializeToXML(Job,filepath);

}

private void tmNotify\_Tick(object sender, EventArgs e)

{

if (!ckbTB.Checked)

return;

Time++;

if (time < Cons.notifyTime)

return;

if (Job == null || Job.ListJob == null || Job.ListJob.Count==0 )

return;

DateTime current = DateTime.Now;

DateTime tomorrow = DateTime.Now.AddDays(1);

//MessageBox.Show(current.Minute.ToString());

if (dtpkDate.Value.Year != current.Year || dtpkDate.Value.Month != current.Month || dtpkDate.Value.Day != current.Day)

return;

for (int i = 0; i < Job.ListJob.Count; i++)

{

//if ( current.Hour >= Job.ListJob[i].ToTime.X && current.Minute >= Job.ListJob[i].ToTime.Y)

if (Job.ListJob[i].ToTime.X \* 3600 + Job.ListJob[i].ToTime.Y \* 60 < current.Hour \* 3600 + current.Minute \* 60

&& Job.ListJob[i].Status != "Done" && Job.ListJob[i].Date.Year == current.Year

&& Job.ListJob[i].Date.Month == current.Month && Job.ListJob[i].Date.Day == current.Day

&& Job.ListJob[i].Job!=null)

{

Job.ListJob[i].Status = "Missed";

}

}

int dem = 0;

for (int i = 0; i < Job.ListJob.Count; i++)

{

if (Job.ListJob[i].Status=="Doing")

{

dem++;

}

}

if (dem==0)

{

return;

}

/\* List<PlanItem> listTodayDone = Job.ListJob.Where

(p => p.Date.Year == current.Year && p.Date.Month == current.Month

&& p.Date.Day == current.Day && PlanItem.list.IndexOf(p.Status) == (int)ePlanItem.Done).ToList();

List<PlanItem> listTodayMissed = Job.ListJob.Where

(p => p.Date.Year == current.Year && p.Date.Month == current.Month

&& p.Date.Day == current.Day && PlanItem.list.IndexOf(p.Status) == (int)ePlanItem.Missed).ToList();

List<PlanItem> listTomorrow = Job.ListJob.Where

(p => p.Date.Year == tomorrow.Year && p.Date.Month == tomorrow.Month

&& p.Date.Day == tomorrow.Day).ToList();\*/

//List<PlanItem> listTen = Job.ListJob.Where(p=>p.Date.Year)

//////////////////////////////////////////////////////////////////////////////////

List<PlanItem> listTodayDoing = Job.ListJob.Where

(p => p.Date.Year == current.Year && p.Date.Month == current.Month

&& p.Date.Day == current.Day && PlanItem.list.IndexOf(p.Status) == (int)ePlanItem.Doing

&& p.FromTime.X \* 3600 + p.FromTime.Y \* 60 <= current.Hour \* 3600 + current.Minute \* 60

&& p.ToTime.X \* 3600 + p.ToTime.Y \* 60 >= current.Hour \* 3600 + current.Minute \* 60).ToList();

string tam = "";

for (int i = 0; i < listTodayDoing.Count; i++)

{

tam += "- "+listTodayDoing[i].Job +"\n";

}

notifyIcon1.ShowBalloonTip(Cons.timeOut, "Lịch công việc",

string.Format("Bạn đang có {0} công việc cần làm: \n", listTodayDoing.Count) + tam, ToolTipIcon.Info);

/\* notifyIcon1.ShowBalloonTip

(Cons.timeOut, "Lịch công việc",

string.Format("Bạn có {0} công việc đang làm, {1} đã hoàn thành," +

" {2} bỏ lỡ trong ngày hôm nay \n" +

"Ngày mai bạn có {3} công việc cần làm",

listTodayDoing.Count,listTodayDone.Count,listTodayMissed.Count,listTomorrow.Count),

ToolTipIcon.Info);\*/

}

private void nmTB\_ValueChanged(object sender, EventArgs e)

{

Cons.notifyTime = (int)nmNotify.Value;

}

private void ckbTB\_CheckedChanged(object sender, EventArgs e)

{

nmNotify.Enabled = ckbTB.Checked;

}

private void Calender\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

private void notifyIcon1\_MouseClick(object sender, MouseEventArgs e)

{

contextMenuStrip1.Show( MousePosition,ToolStripDropDownDirection.AboveRight);

}

private void tsmnJob\_Click(object sender, EventArgs e)

{

WindowState = FormWindowState.Normal;

this.ShowInTaskbar = true;

Plan plan = new Plan(dtpkDate.Value, Job);

plan.Show();

}

private void tsmnExit\_Click(object sender, EventArgs e)

{

Application.Exit();

}

}

}

**KẾT LUẬN**

1. **Tổng kết báo cáo**

Nhóm đã giải quyết được đa số các vấn đề của bài tập lớn. Đã hoàn thành khá tốt chương trình bằng ngôn ngữ C# Windows Forms.Tuy đã nghiên cứu và tìm hiểu nhưng vẫn chưa thể tối ưu và vẫn còn thiếu sót rất nhiều

1. **Ưu điểm và nhược điểm của chương trình**

* **Ưu điểm:**
* Chương trình dễ sử dụng,giao diện thân thiện với người dùng.
* Có những chức năng cần thiết dễ dàng cho người sử dụng thao tác như : Thêm-sửa-xóa công việc, thông báo công việc, lưu trữ công việc,…
* **Nhược điểm:**
* Vẫn chưa chú trọng về đồ họa giao diện, giao diện vẫn còn đơn giản
* Code chương trình còn dài,chưa thể tối ưu, vẫn còn thiết sót nhiều tính năng

1. **Hướng phát triển**

Xây dựng các tính năng như giao diện,thêm nhiều thuộc tính hơn,cần có thêm nhiều chức năng và đầy đủ hơn nữa để người dùng dễ dàng sử dụng,…

**TÀI LIỆU THAM KHẢO**

1. Bài giảng “ **Lập trình Windows với C#.NET**”,khoa CNTT,ĐH HHVN .
2. **Lập trình Windows C#,** Nhà xuất bản KHKT, 2015
3. <http://luanvan.co/luan-van/de-tai-thiet-ke-phan-mem-quam-ly-lich-lam-viec-ca-nhan-39180/>