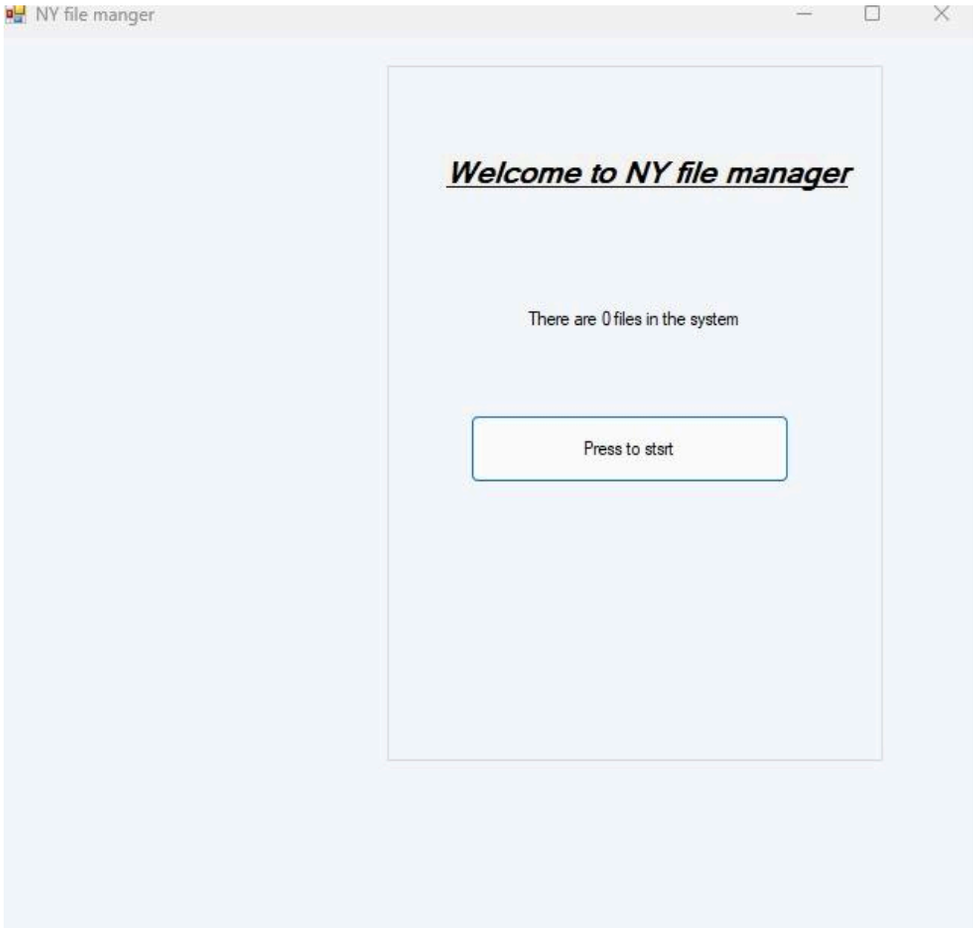
GUI 4

נועם ברחד 314868399

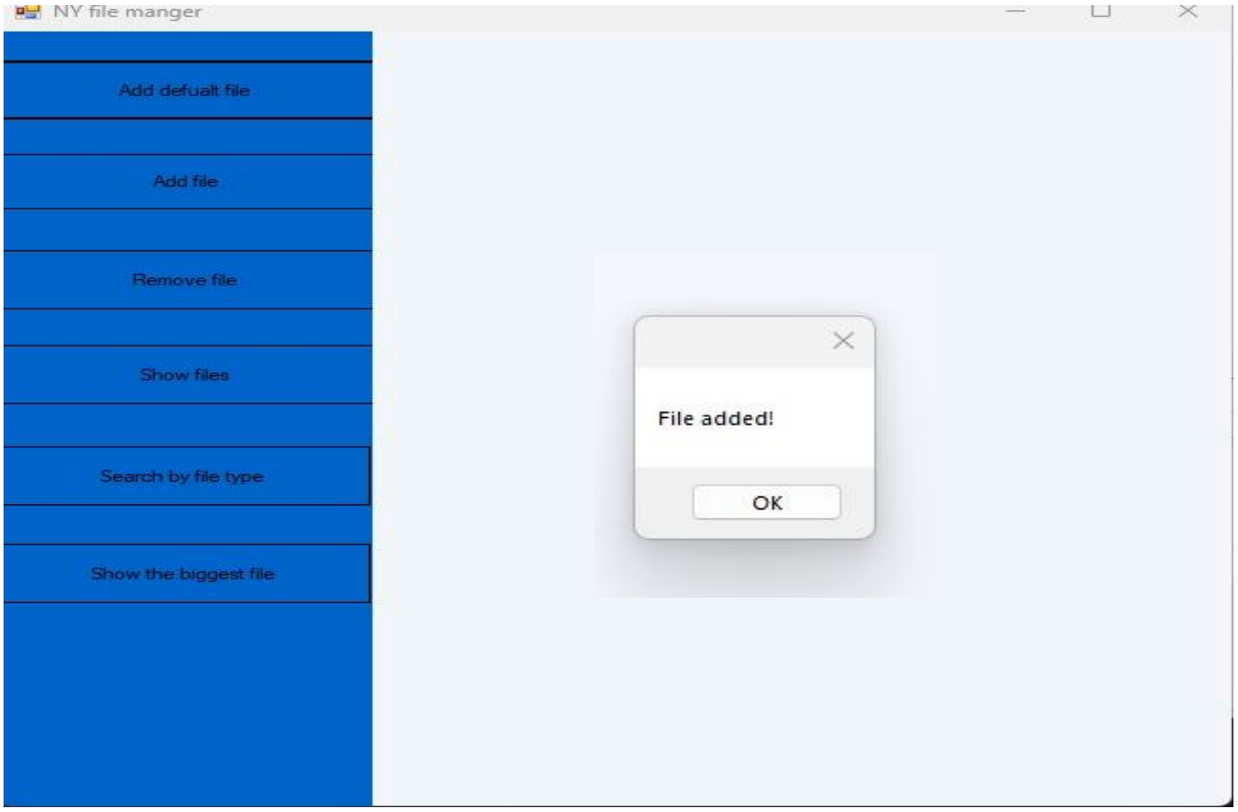
ירדן שוורץ 316135904

שאלה 1.

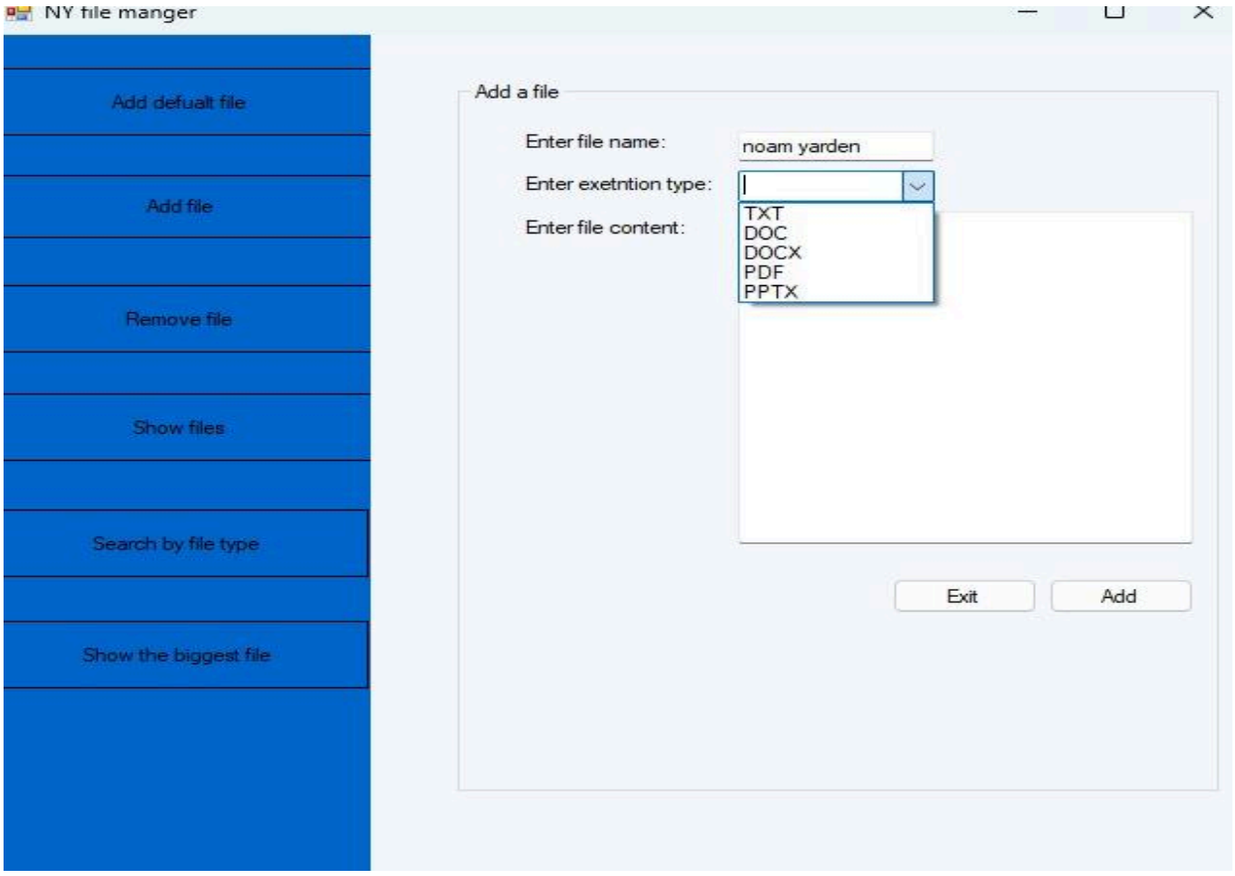
מסך הפתיחה:



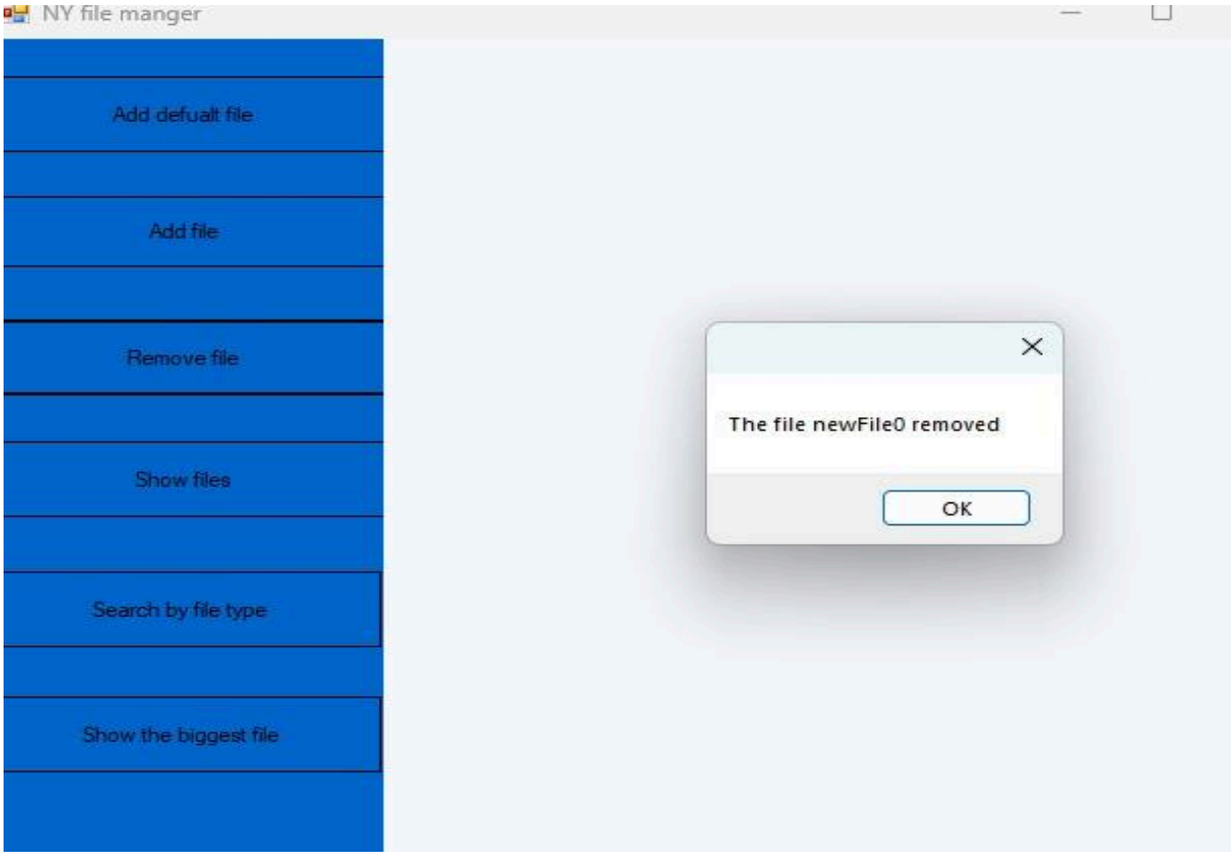
לחיצה על הכפתור תפתח תפריט ו כאשר נלחץ על Add default file יופיע על המסך :



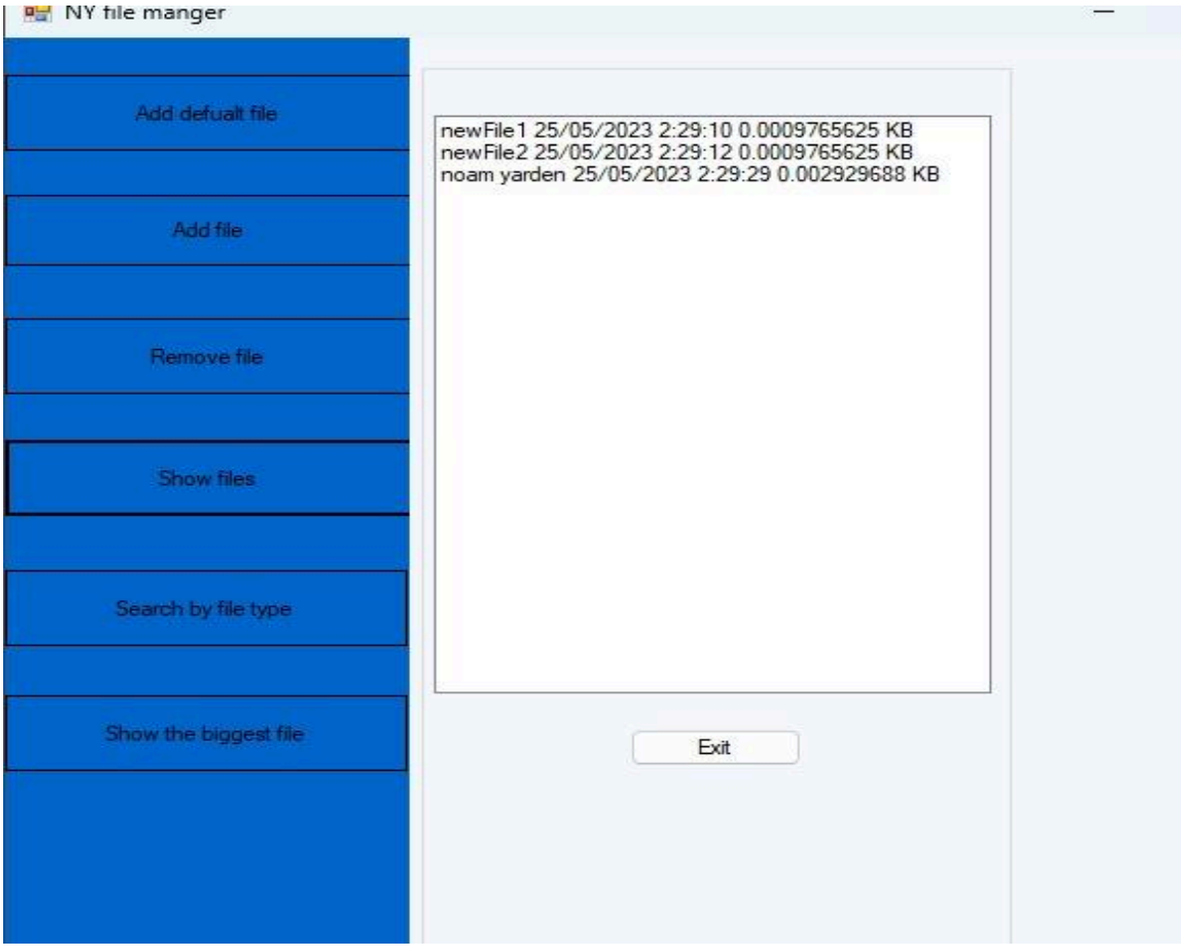
כאשר נלחץ על Add file יופיע על המסך ונוכל להוסיף קובץ ייחודית אם המשתמש לא יזין את הנתונים תתקבל הודעת שגיאה :



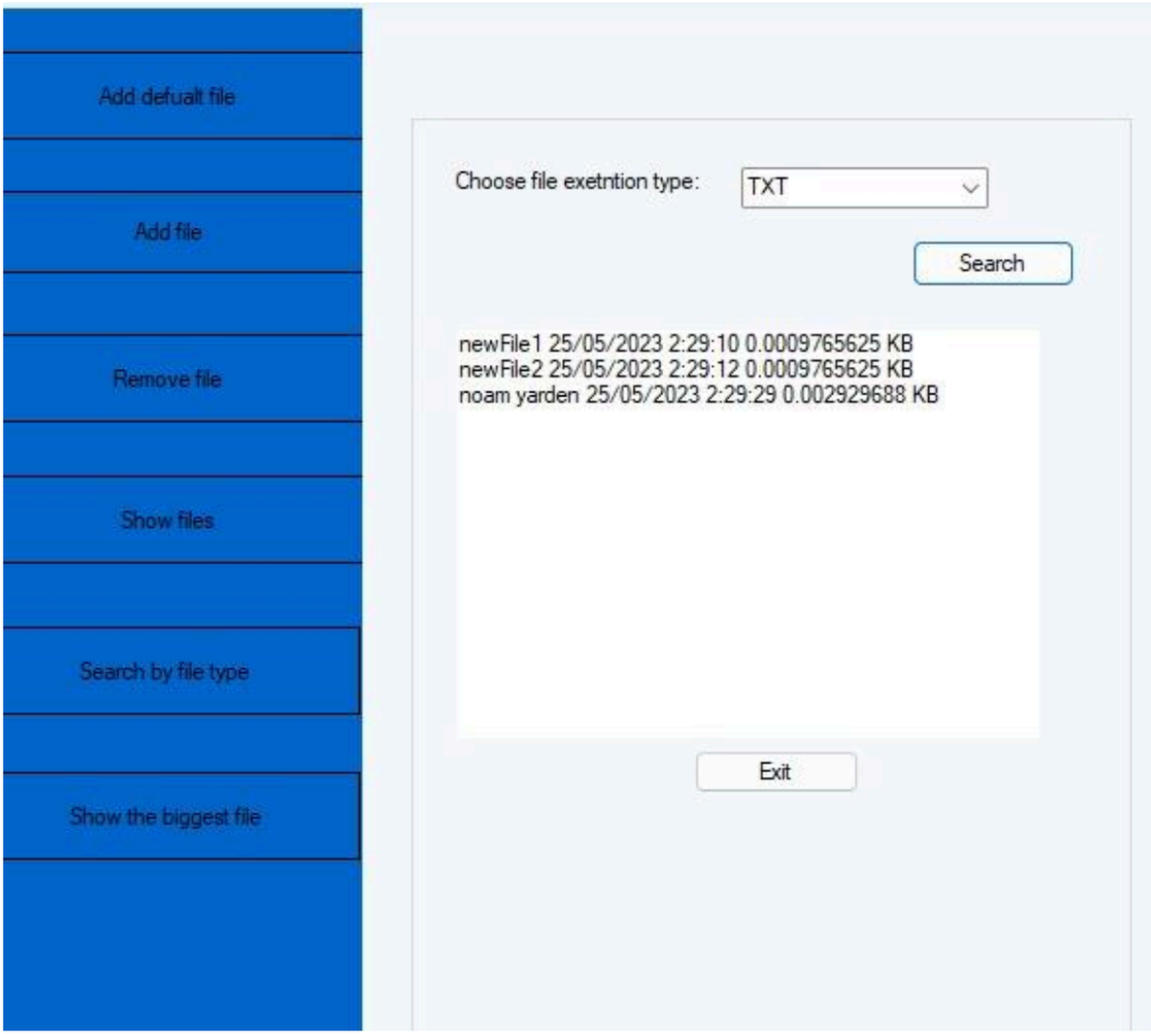
לחיצה על Remove file תמחק את הקובץ הראשון שנשמר במערכת ויופיע הודעה מתאימה.



כאשר נלחץ על Show files תפתח רשימת הקבצים שנשמרו :



Search file by type יחפש לנו קבצים מהרשימה לפי סוג הקובץ שנבחר:


Add defualt file
Add file
Remove file
Show files
Search by file type
Show the biggest file
Choose file exetntion type:
TXT
Search
newFile1 25/05/2023 2:29:10 0.0009765625 KB
newFile2 25/05/2023 2:29:12 0.0009765625 KB
noam yarden 25/05/2023 2:29:29 0.002929688 KB
Exit

Show the biggest יחזיר לנו את הקובץ הגדול ביותר:

כאשר הרשימה ריקה כול אחד מהפקדים יחזיר הודעה מתאימה.

FORM

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Imaging;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Gui_4
```

```
{
    public partial class FrmFirstWindow : Form
    {
        private QueueFiles files;
        private QueueFiles Qlist = new QueueFiles();
        private FileTypeExtension type;

        public FrmFirstWindow()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            HideGrb();
            if (Qlist.IsEmpty())
            {
                lblNoFiels.Visible = true;
                grStart.Visible = true;
            }



        }

        private void btnDefualtF_Click(object sender, EventArgs e)
        {
            DataFile file1 = new DataFile();
            Qlist.Enqueue(file1);
            UploadFileToTheList();
            MessageBox.Show("File added!");
        }
```

```
private void btbAddF_Click(object sender, EventArgs e)
{
    grbAdd.Visible = true; grbAdd.Enabled = true;
}

private void btnStart_Click(object sender, EventArgs e)
{
    panel1.Visible = true;
    lblNoFiels.Visible = false;
    lblstart.Visible = false;
    btnStart.Visible=false;
    HideGrb();

}

private void btnRemoveF_Click(object sender, EventArgs e)
{
    HideGrb();
    if (Qlist.IsEmpty()) { MessageBox.Show("No files at the list"); }
    else
    {
        DataFile dfile = Qlist.Dequeu();
        UploadFileToTheList();
        string rF = dfile.GetFileName();
        MessageBox.Show("The file " + rF + " removed");
    }
}

private void btnAddFile_Click(object sender, EventArgs e)
{
```

```
        if (string.IsNullOrEmpty(txtFname.Text))
        {
            MessageBox.Show("Please fill the file name ");
            return;
        }
        if(string.IsNullOrEmpty(txtFcontent.Text))
        { MessageBox.Show("Please fill file content");
            return;
        }
         string fname  = txtFname.Text;
        string fcontent = txtFcontent.Text;
        if(cmbType.SelectedItem != null) {
            { FileTypeExtension fileType = (FileTypeExtension)Enum.Parse(typeof(FileTypeExtension),
cmbType.SelectedItem.ToString());
                DataFile file = new DataFile(fname, fcontent, fileType);
                if (file.GetFileName()==fname)
                {
                    Qlist.Enqueue(file);
                    UploadFileToTheList();
                    MessageBox.Show("File " + fname + " added!");
                }
            }
        }
        txtFcontent.Clear();
        txtFname.Clear();
        cmbType.SelectedIndex = -1;


    }
    private void UploadFileToTheList()
    {
        filesList.Items.Clear();
```

```
    foreach (DataFile file in Qlist.CopyToArry())
    {
        filesList.Items.Add(file.Dir());
    }
}

private void btnShowF_Click(object sender, EventArgs e)
{
    if (Qlist.IsEmpty())
    {
        MessageBox.Show("No files at the list");
    }
    else
    {
        HideGrb();
        grbFileList.Visible = true;
    }

}
private void HideGrb()
{
    grStart.Visible = false;
    grbFileList.Visible = false;
    grbAdd.Visible = false;
    grbSearch.Visible = false;
    grbBig.Visible = false;
}

private void btnExitAddFile_Click(object sender, EventArgs e)
{
    HideGrb();
```

```
    }

    private void btnExitList_Click(object sender, EventArgs e)
    {
        HideGrb();
    }

    private void btnSearchF_Click(object sender, EventArgs e)
    {
        grbSearch.Visible=true;
    }

    private void cmbSearchType_SelectedIndexChanged(object sender, EventArgs e)
    {
        if (cmbSearchType.SelectedIndex != null)
        {
type=(FileTypeExtension)Enum.Parse(typeof(FileTypeExtension),cmbSearchType.SelectedItem.ToStri
ng());
        }
    }

    private void btnSearch_Click(object sender, EventArgs e)
    {
        if (Qlist.IsEmpty())
        {
            HideGrb();
            MessageBox.Show("No files at the list");
        }
        else
        {
            lstSearchType.Items.Clear();
```

```
        lstSearchType.Visible = true;
        DataFile[] temp = Qlist.SearchFileByType(type);
        foreach (DataFile f in temp)
        {
            lstSearchType.Items.Add(f.Dir());
        }
    }
}

private void btbExitSearch_Click(object sender, EventArgs e)
{
    HideGrb();
}

private void btnShowBF_Click(object sender, EventArgs e)
{
    if (Qlist.IsEmpty())
    {
        MessageBox.Show("No files at the list");
    }
    else
    {
        DataFile Big = Qlist.BigFile();
        if (Big != null)
        {
            grbBig.Visible = true;
            txtBig.Text = Big.Dir();
        }
    }
}
```

```
        private void btnExitBig_Click(object sender, EventArgs e)
        {
            HideGrb();
        }

        private void lblNoFiels_Click(object sender, EventArgs e)
        {

        }
    }
}
```

CLASS DATAFILE

```
namespace Gui_4
{
    public enum FileTypeExtension { TXT = 1, DOC, DOCX, PDF, PPTX };
   public class DataFile
    {
        private string FileName;
        private DateTime LastUpadateTime;
        private string Data;
        private FileTypeExtension type;
        public static int counter = 0;

        public string GetFileName()
        {
            return FileName;
        }
        public string GetData()
        {
            return Data;
        }
```

```
public DateTime GetTime()
{
    return LastUpadateTime;
}
public FileTypeExtension GetFileType() { return type; }
public FileTypeExtension setFileType() { return this.type; }

public void SetFilename(string FileName)
{
    for (int j = 0; j < FileName.Length; j++)
    {
        if (FileName[j] >= '!' && FileName[j] <= '/' || FileName[j] >= ':' && FileName[j] <= '@' ||
FileName[j] >= '[' && FileName[j] < 'a')
        {

            return;
        }

    }
    this.FileName = FileName;
}
public void SetData(string Data)
{
    this.Data = Data;
}
public void SetTime()
{
    LastUpadateTime = DateTime.Now;
}
public DataFile(string FileName, string Data, FileTypeExtension type)
{
```

```
    this.FileName = FileName;
    this.Data = Data;
    this.SetFilename(FileName);
    this.LastUpadateTime = DateTime.Now;
    this.type = type;



}
public DataFile() : this("newFile" + counter, " ", FileTypeExtension.TXT)
{

    counter++;
}
public DataFile(DataFile other)
{
    FileName = "Copy of " + other.FileName;
    Data = other.Data;
    SetTime();
}
public int GetSize()
{

    int size;
    size = Data.Length;

    return size;
}
public string Dir()
{
    return FileName + " " + LastUpadateTime.ToString()+ " "+((float)this.GetSize()/1024)+" KB ";
```

```
        }
    }
}
```

CLASS CompareFiles

```
namespace Gui_4
{
    public static class CompareFiles
    {
        public static bool EqualFiles(DataFile file1, DataFile file2)
        {
            if (file1.GetFileName() != file2.GetFileName() || file1.GetData() != file2.GetData())
            {
                return false;
            }
            return true;
        }
        public static int CompareSize(DataFile file1, DataFile file2)
        {
            int size1 = file1.GetSize();
            int size2 = file2.GetSize();

            if (size1 > size2)
            {
                return 1;
            }
            else if (size2 > size1)
            {
                return -1;
            }
            else
            {
```

```
            return 0;
        }
    }
  }
}
```

Class QueueFiles

```
namespace Gui_4
{
  public class QueueFiles
  {

    private int index;
    private DataFile[] arr;
    public QueueFiles()
    {
      arr = new DataFile[0];
      index = -1;
    }
    public bool IsEmpty()
    {
      if (index == -1)
      {

        return true;
      }
      return false;
    }
    public void Enqueue(DataFile fille)
    {
      try
      {
```

```
        bool fexists = false;
        foreach (var item in arr)
        {
            if (CompareFiles.EqualFiles(item, fille))
            {
                fexists = false;
                break;
            }
        }
        if (fexists) { return; }
        index++;
        if (index == arr.Length)
        {
            Array.Resize(ref arr, arr.Length + 1);
        }
        arr[index] = fille;
    }
    catch (Exception ex) { MessageBox.Show("Error " + ex.Message); }
}
public DataFile Dequeu()
{
    if (IsEmpty())
    {
        return null;
    }
    DataFile DelFile = arr[0];
    for (int i = 1; i <= index; i++)
    {
        arr[i - 1] = arr[i];
    }
    index--;
```

```
    return DelFile;
}
public DataFile BigFile()
{



    if (index == 0)
    {

        return arr[0];
    }
    QueueFiles TempQ = new QueueFiles();
    DataFile MaxFile = arr[0];
    foreach (DataFile f in arr)
    {
        if (CompareFiles.CompareSize(f, MaxFile) > 0)
        {
            MaxFile = f;
        }
        TempQ.Enqueue(f);
    }
    if (!TempQ.IsEmpty())
    {
        Enqueue(TempQ.Dequeu());
    }
    return MaxFile;
}
public void PrintQueue()
{
    if (IsEmpty())
    {
```

```
            return;
        }

        for (int i = 0; i <= index; i++)
        {
            arr[i].Dir();
        }
    }
    public DataFile[] SearchFileByType(FileTypeExtension type)
    {

        List<DataFile> list = new List<DataFile>();
        foreach (DataFile f in arr)
        {
            if (f.GetFileType() == type) { list.Add(f); }
        }
        return list.ToArray();

    }
    public DataFile[] CopyToArry() {
        DataFile[] copyToArray = new DataFile[index + 1];
        Array.Copy(arr, copyToArray, index+1);
        return copyToArray;
    }



  }
}
```