מטלה מספר 4

GUI

מגישים :

גלי לוי 318173283

יובל רופא 205823180

שאלה 1

form

```
using System;
using System.Collections;
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 static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace WindowsFormsApp1
{
    public partial class Form1 : Form
    {
        QueueFiles queue = new QueueFiles();
        public Form1()
        {
            InitializeComponent();


        }
        private void Form1_Load(object sender, EventArgs e)
        {
            Fillmenu();
            Filltypecombo();

        }
        public void Fillmenu ()
        {
            LbxMenu.Items.Add("1 - add a default file to the queue");
            LbxMenu.Items.Add("2 - add a new file to the queue");
            LbxMenu.Items.Add("3 - remove file from the queue");
            LbxMenu.Items.Add("4 - print the queue");
            LbxMenu.Items.Add("5 - print all the files with the same
type");
            LbxMenu.Items.Add("6 - print the biggest file from the
queue");
            LbxMenu.Items.Add("7 - finsh and shoutdown");
        }
        public void Filltypecombo()
        {
            CbxType.Items.Add(FileTypeExtension.TXT);
            CbxType.Items.Add(FileTypeExtension.DOC);
            CbxType.Items.Add(FileTypeExtension.DOCX);
            CbxType.Items.Add(FileTypeExtension.PDF);
            CbxType.Items.Add(FileTypeExtension.PPTX);
            CbxTypeChooseTypePrint.Items.Add(FileTypeExtension.TXT);
            CbxTypeChooseTypePrint.Items.Add(FileTypeExtension.DOC);
            CbxTypeChooseTypePrint.Items.Add(FileTypeExtension.DOCX);
            CbxTypeChooseTypePrint.Items.Add(FileTypeExtension.PDF);
            CbxTypeChooseTypePrint.Items.Add(FileTypeExtension.PPTX);
        }

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

```
            BtnRemove.Visible = false;
            LblNoFile.Visible = false;
            GbxAddFile.Visible = false;
            TbxPrint.Visible = false;
            GbxType.Visible = false;
            BtnChooseType.Enabled = false;
            TbxPrint.Text = "";

            int x =  LbxMenu.SelectedIndex;
            switch(x)
            {
                case 0:
                    {
                        bool check = true;
                        DataFile file = new DataFile();
                        check = queue.Enqueue(file);
                        if (check)
                        {
                            MessageBox.Show("The file added
succesfully!");

                            break;
                        }
                        MessageBox.Show("The file already exist");

                        break;
                    }
                case 1:
                    {

                        GbxAddFile.Visible = true;

                        break;
                    }
                case 2:
                    {

                        BtnRemove.Visible = true;

                        break;
                    }
                case 3:
                    {
                        TbxPrint.Visible = true;
                        TbxPrint.Text = queue.PrintQueue();

                        break;
                    }
                case 4:
                    {
                        GbxType.Visible = true;


                        break;
                    }
                case 5:
                    {
                        TbxPrint.Visible = true;
                        TbxPrint.Text = queue.BigFile().Dir();

                        break;
                    }
                case 6:
```

```
                {
                    Close();
                    break;
                }

            }
        }

        private void BtnAddFile_Click(object sender, EventArgs e)
        {
            if (TbxName.Text=="" || TbxData.Text=="" ||
CbxType.SelectedIndex==-1)
            {
                MessageBox.Show("Please insert all the details");
            }
            else
            {
                bool check = true;
                object temp = CbxType.SelectedItem;
                string s = Convert.ToString(CbxType.SelectedItem);
                FileTypeExtension type =
(FileTypeExtension)Enum.Parse(typeof(FileTypeExtension),s);
                DataFile addfile = new DataFile(TbxName.Text,
TbxData.Text,type);
                check=queue.Enqueue(addfile);
                GbxAddFile.Visible = false;
                BtnRemove.Visible = false;
                if (check)
                {
                    MessageBox.Show("The file added succesfully!");
                    LblNoFile.Visible = false;
                    TbxName.Text = "";
                    TbxData.Text = "";
                    CbxType.ResetText();
                    GbxAddFile.Visible = false;
                    BtnRemove.Visible = false;
                }
                else
                {
                    MessageBox.Show("The file already exist");
                    LblNoFile.Visible = false;

                }
            }
        }

        private void BtnRemove_Click(object sender, EventArgs e)
        {
            if (queue.IsEmpty())
            {
                MessageBox.Show("The Arry is empty");
            }
            else
            {
                queue.Dequeue();
                MessageBox.Show("The file has been removed ");
            }
        }

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

```
            string s =
Convert.ToString(CbxTypeChooseTypePrint.SelectedItem);
            FileTypeExtension type =
(FileTypeExtension)Enum.Parse(typeof(FileTypeExtension), s);
            DataFile[] temp = queue.SearchFileByType(type);
            if (temp == null)
            {
                TbxPrint.Text = "There is no file with the type you
chose";
            }
            else
            {
                QueueFiles ListType = new QueueFiles(temp, temp.Length);
                TbxPrint.Text = ListType.PrintQueue();

            }
            GbxType.Visible = false;
            TbxPrint.Visible = true;
        }

        private void CbxTypeChooseTypePrint_SelectedIndexChanged(object
sender, EventArgs e)
        {
            BtnChooseType.Enabled = true;
        }

        private void LbxMenu_StyleChanged(object sender, EventArgs e)
        {

        }
    }
}
```

DataFile

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WindowsFormsApp1
{
    internal class DataFile
    {

        private string FileName;
        private DateTime lastUpadateTime;
        private string Data;
        readonly FileTypeExtension type;
        static int counter=0;

        public string GetFileName()
        {

            return FileName;
        }

        public FileTypeExtension getType()
        {
            return type;
        }


        public void SetFileName(string fName)
        {

            for (int i = 0; i < fName.Length; i++)
            {
                if (fName[i] == '<' || fName[i] == '>' || fName[i] == '?'
|| fName[i] == '*' || fName[i] == ':' || fName[i] == '\'' || fName[i] ==
'/')
                {
                    Console.WriteLine("A file name can't contain any of
the following characters (<,>,?,*,:,/,\'), please try again...");
                    return;
                }
            }
            FileName = fName;
        }
        public string GetData()
        {
            return Data;
        }
        public void SetData(string Data1)
        {
            Data = Data1;
        }

        public void SetTime()
        {
            lastUpadateTime = DateTime.Now;
        }
        public DateTime GetTime()
        {
```

```
            return lastUpadateTime;
        }

        public DataFile(string FileName, string Data, FileTypeExtension
type)  // 1בנאף
        {
            this.type = type;
            SetFileName(FileName);
            SetData(Data);
            SetTime();
        }
        public DataFile() : this("sameFile"+counter,
"",FileTypeExtension.TXT)  // 2 בנאי
        {

        }

        public DataFile(DataFile other)  // 3 בנאי
        {

            this.Data = other.Data;
            this.lastUpadateTime = other.lastUpadateTime;
            this.FileName = other.FileName;
            this.type = other.type;
        }
        public int GetSize()
        {
            return Data.Length;
        }
        public string Dir()
        {
            int FileSize = GetSize() / 1024;
            return lastUpadateTime + " " + FileSize + " KB " + FileName +
"\n" ;
        }
    }
}
```

Program

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp1
{
    enum FileTypeExtension
    { TXT = 1, DOC, DOCX, PDF, PPTX }
    internal static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

QueueFiles

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WindowsFormsApp1
{
    internal class QueueFiles
    {
        DataFile[] arrFile;
        int index;

        public QueueFiles()
        {
            arrFile = new DataFile[0];
            index = -1;
        }

        public QueueFiles(DataFile[] arr,int index)
        {
            this.arrFile = arr;
            this.index = index;
        }
        public bool IsEmpty()
        {
            if (arrFile.Length == 0)
            {
                Console.WriteLine("The array is empty\n");
                return true;
            }
            return false;
        }

        public bool Enqueue(DataFile x) /מוסיף אובייקט לרשימה/
        {
            bool check = false;
            if (index > 0)
            {
                for (int i = 0; i < arrFile.Length; i++)
                {
                    if (CompareFiles.EqualFiles(arrFile[i], x))
                    {
                        check = true;
                        return false;
                    }
                }

            }
            if (check==false)
            {
                if (index == -1)
                {
                    index++;
                }
                DataFile[] temp = new DataFile[arrFile.Length + 1];
                for (int i = 0; i < arrFile.Length; i++)
                {
                    temp[i] = arrFile[i];
                }
                temp[index] = x;
```

```
            arrFile = temp;
            index++;
            return true;
        }
        return false;


    }

    public DataFile Dequeue() /שומרת את האיבר הראשון ומסירה אותו מ
מהרשימה
    {
        if (IsEmpty())
        {
            Console.WriteLine("The array is empty\n");
            return null;
        }
        DataFile temp = arrFile[0];
        DataFile[] arr2 = new DataFile[arrFile.Length - 1];
        for (int i = 0; i < index - 1; i++)
        {
            arr2[i] = arrFile[i + 1];
        }
        index--;
        arrFile = arr2;
        return temp;
    }

    public DataFile BigFile()
    {
        if (IsEmpty())
        {
            return null;
        }
        else
        {
            QueueFiles tempQueue = new QueueFiles();
            tempQueue.arrFile = arrFile;
            tempQueue.index = index;
            if (index==1)
            {
                Console.WriteLine("The array contain only 1 File\n");
                return tempQueue.Dequeue();
            }

            DataFile tempFile = new DataFile();
            DataFile maxFile = new DataFile();
            maxFile= tempQueue.Dequeue();
            while (tempQueue.index > 0)
            {
             tempFile = tempQueue.Dequeue();

                if (CompareFiles.CompareSizeFiles(maxFile,tempFile)==-
1)
                {
                    maxFile = tempFile;
                }

            }

            return maxFile;
        }
```

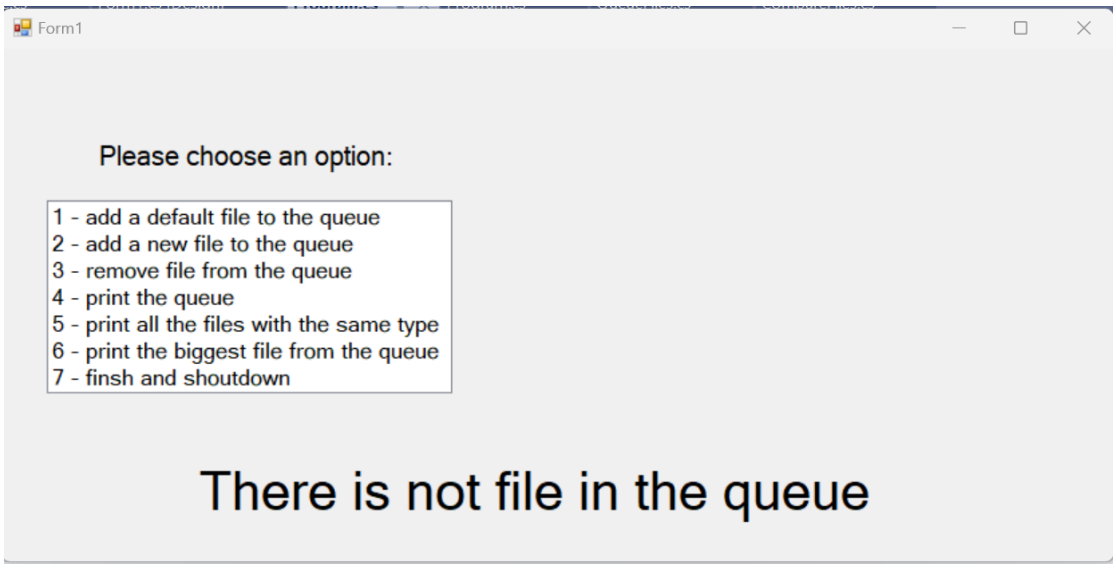
```
        }

        public string PrintQueue()
        {
            string s = "";
            if (IsEmpty())
            {
                 s = "The array is empty";

            }
            else
            {
                QueueFiles tempQueue = new QueueFiles();
                tempQueue.arrFile = arrFile;
                tempQueue.index = index;


                while(tempQueue.index>0)
                {
                    s += tempQueue.Dequeue().Dir() + "\r\n" ;
                }

            }
                return s ;
        }

        public DataFile[] SearchFileByType(FileTypeExtension type)
        {
            if (IsEmpty())
            {
                Console.WriteLine("The array is empty\n");
                return null;
            }

            QueueFiles temp = new QueueFiles();
            QueueFiles sameType = new QueueFiles();
            DataFile tempFile = new DataFile();
            temp.arrFile = arrFile;
            temp.index = index;
            while (temp.index>0)
            {
                tempFile = temp.Dequeue();
                if (tempFile.getType()==type)
                {
                    sameType.Enqueue(tempFile);
                }
            }
            if (sameType.index==-1)
            {
                Console.WriteLine("There is no file with the same
type\n");
                return null;
            }
            return sameType.arrFile;
        }


    }
}
```

CompareFiles

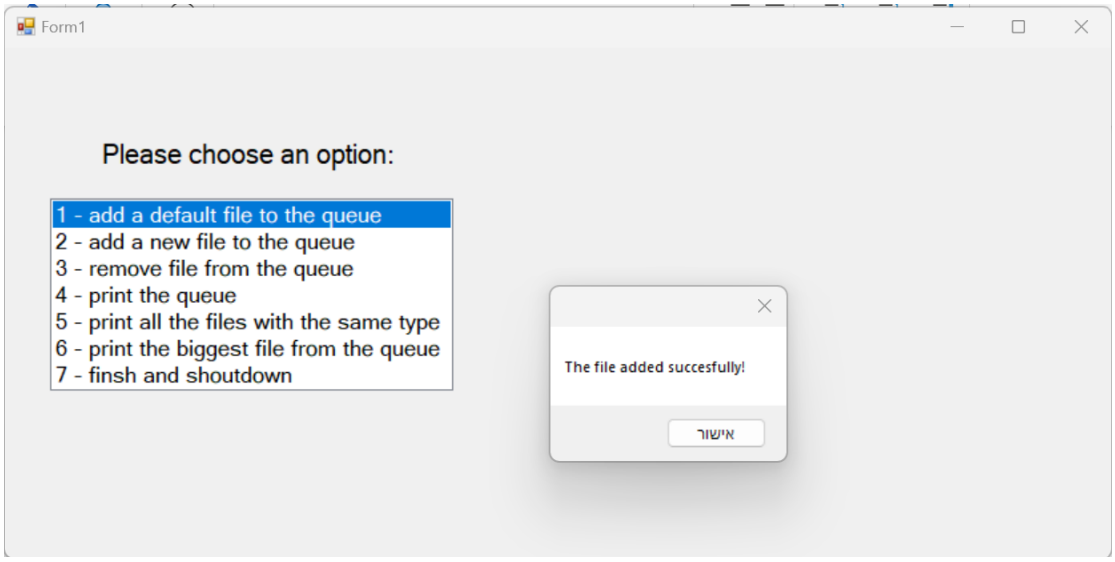
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Cryptography.X509Certificates;
using System.Text;
using System.Threading.Tasks;

namespace WindowsFormsApp1
{
    internal static class CompareFiles
    {
        public static bool EqualFiles(DataFile x,DataFile y)
        {
            if (x.GetFileName()==y.GetFileName() &&
x.GetData()==y.GetData())
            {
                return true;
            }
            return false;
        }


        public static int CompareSizeFiles(DataFile x, DataFile y)
        {
            if (x.GetSize()>y.GetSize())
            {
                return 1;
            }
            if (x.GetSize()<y.GetSize())
            {
                return -1;
            }
            return 0;
        }



    }
}
```

פלטים :

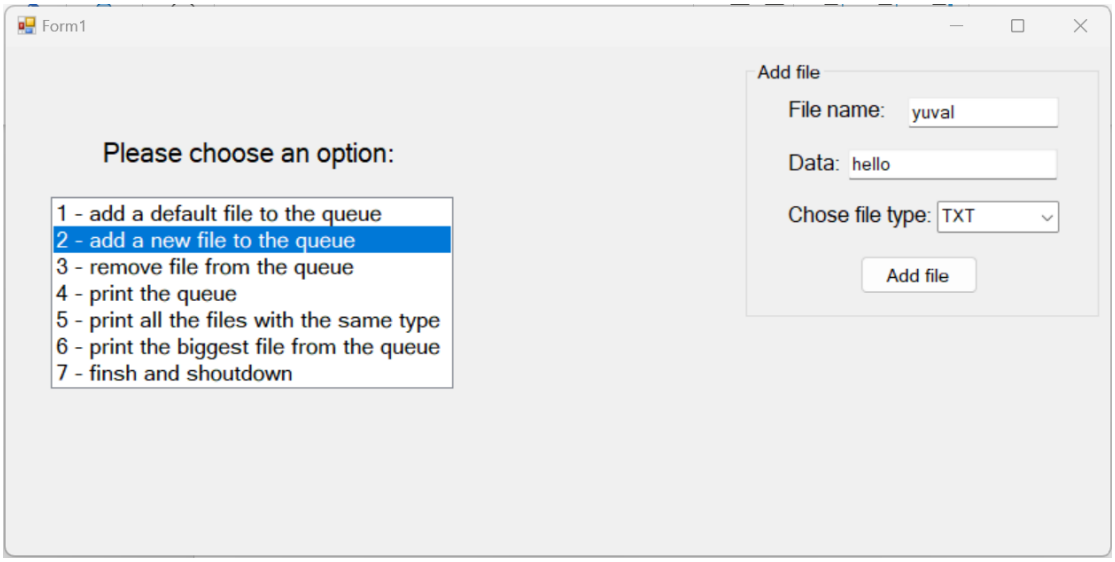
מסך כניסה -



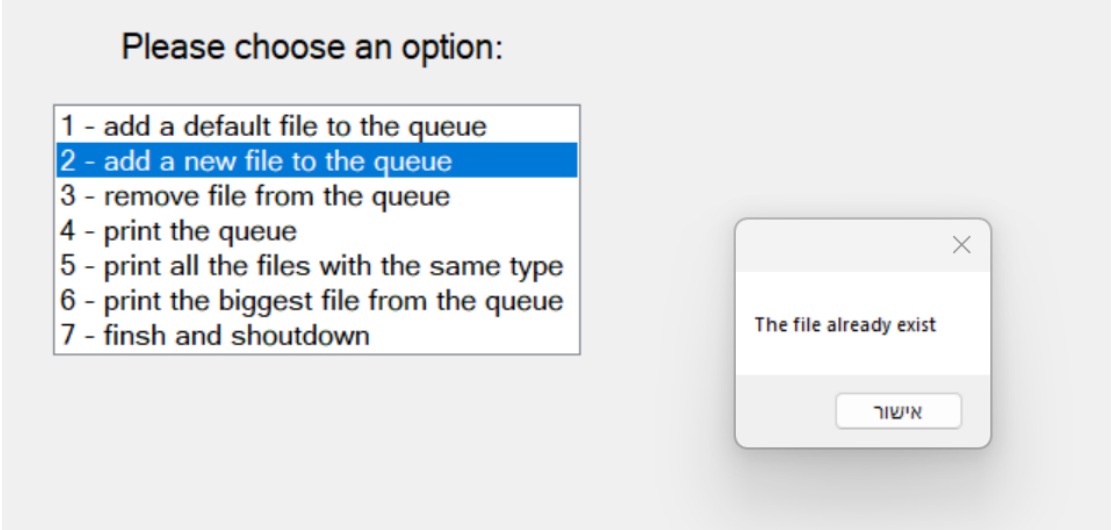
אופציה ראשונה-הוספת קובץ דיפולטיבי



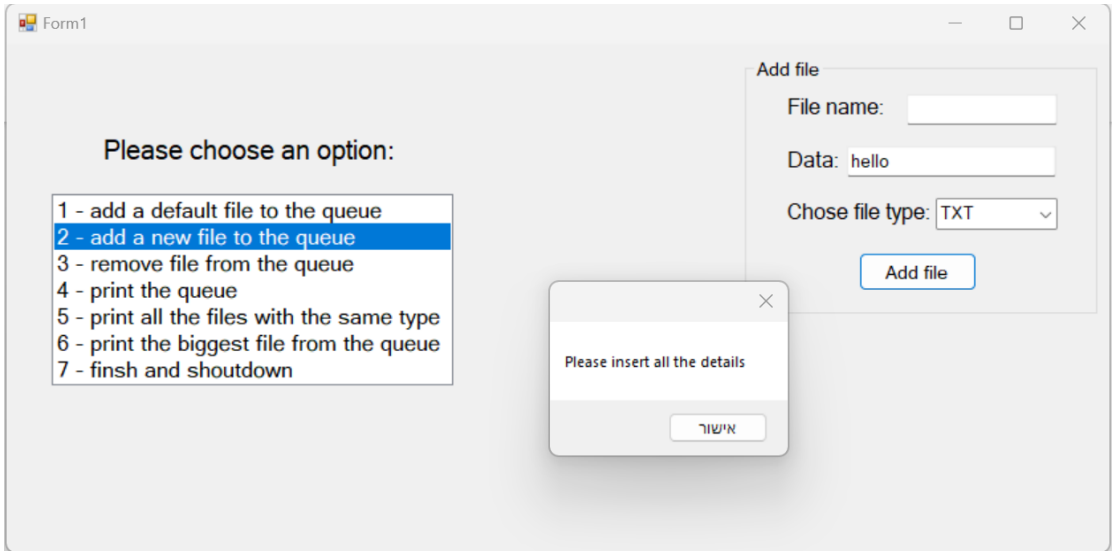
אופציה 2 – הוספת קובץ חדש



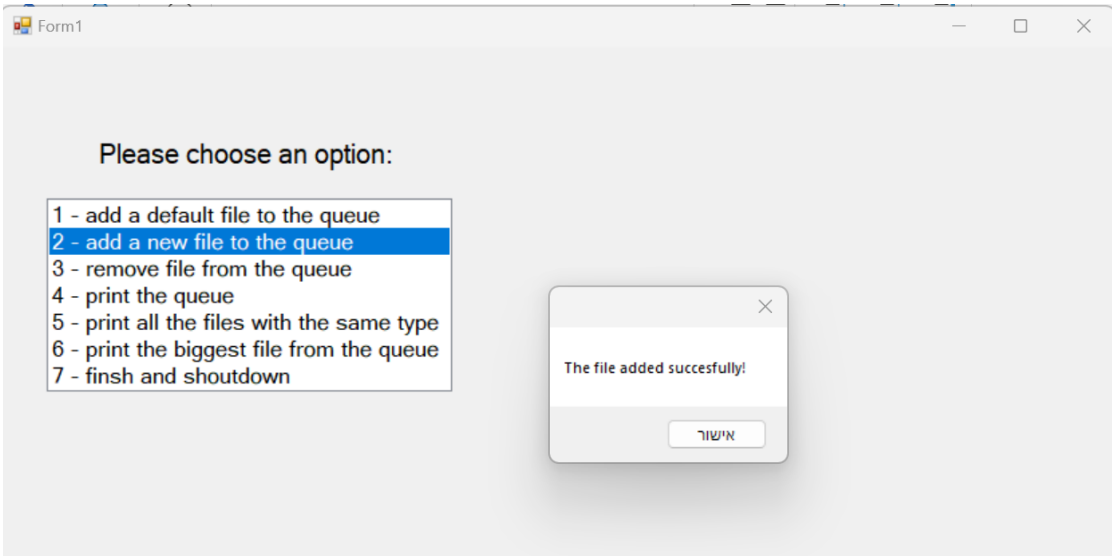
הוספת קובץ קיים



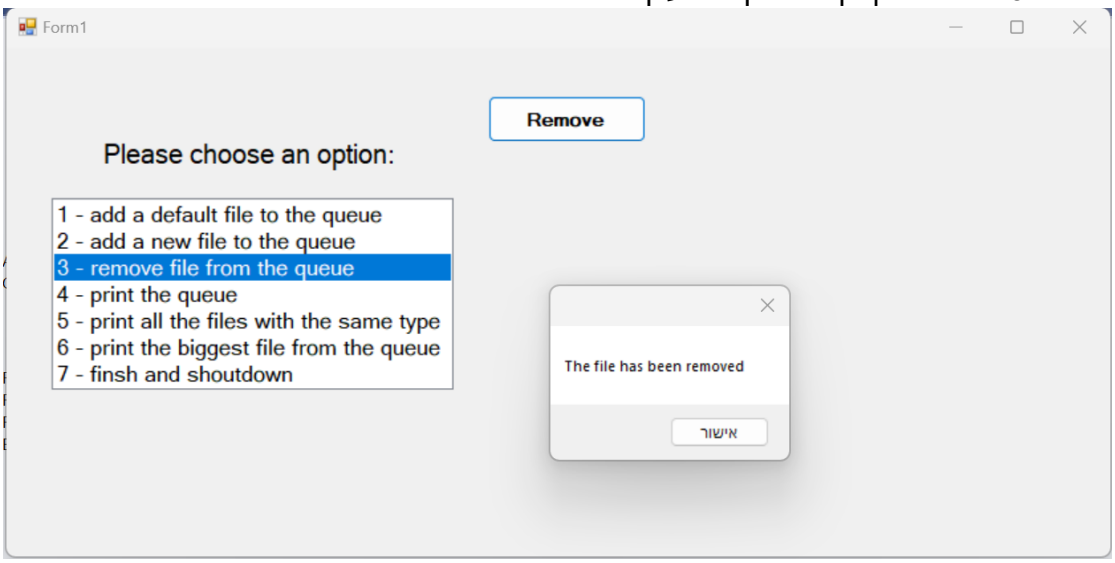
הוספה שנכשלה בעקבות חוסר בפרטים



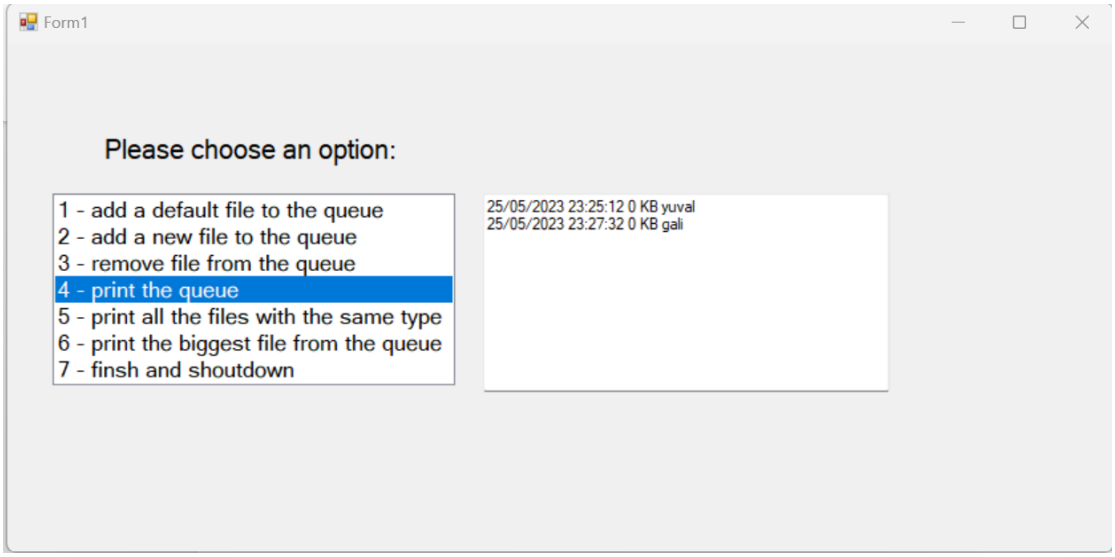
הוספת קובץ בהצלחה



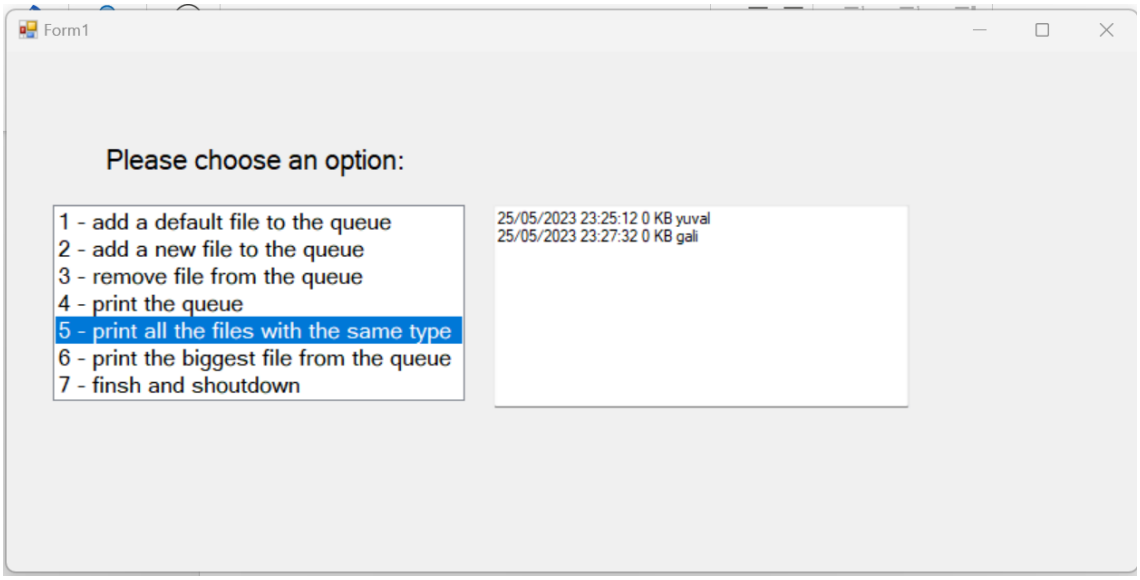
אופציה 3 - הסרת הקובץ הראשון מהמערך



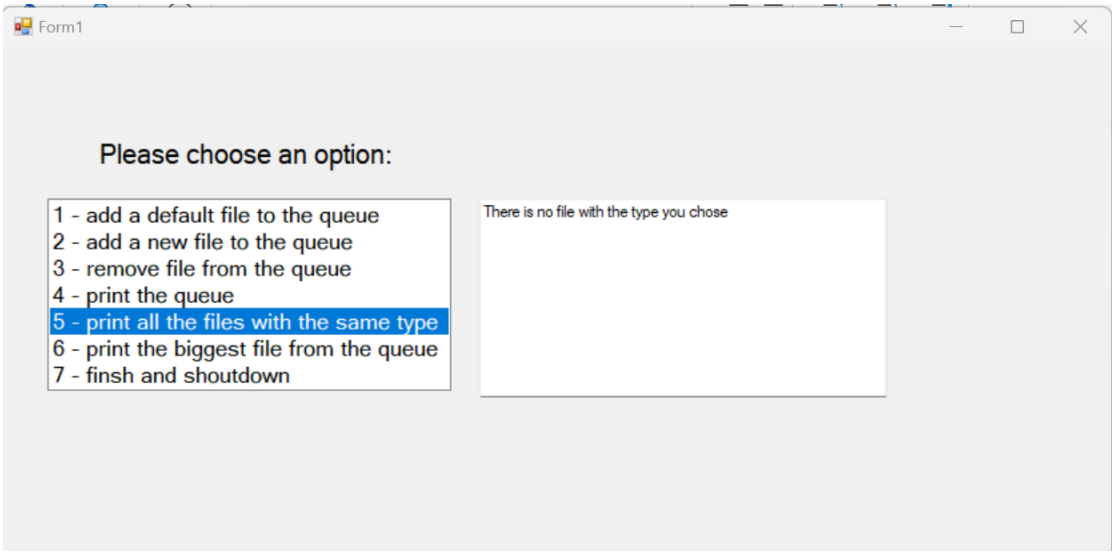
אופציה 4 - הדפסת מערך הקבצים



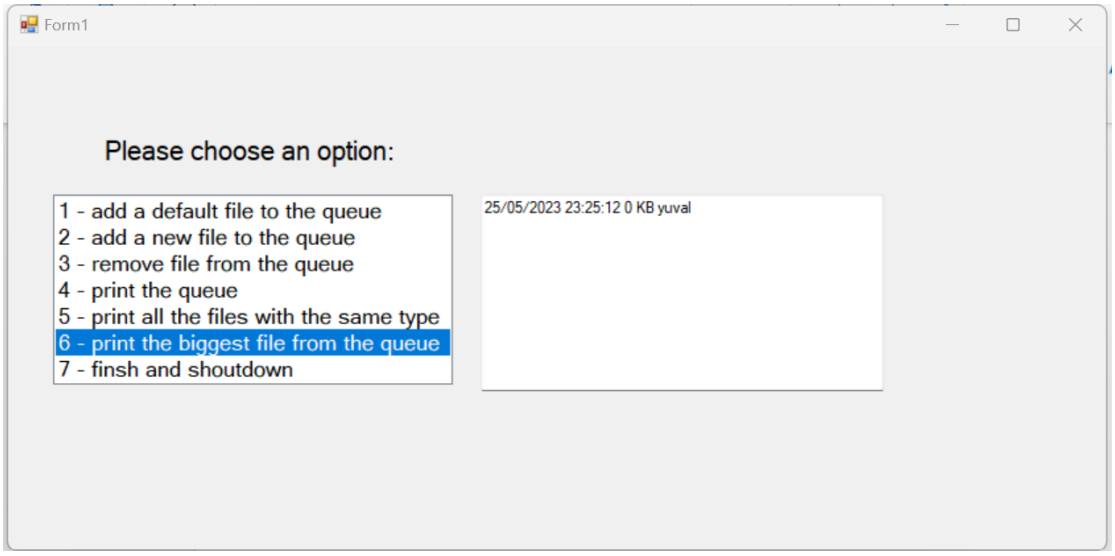
אופציה 5- הדפסת קובץ מסוג מסוים



בחירת סוג שלא קיים



אופציה 6- הדפסת הקובץ הגדול ביותר (לפי אורך DATA)



אופציה 7 – יוצאת מהמערכת.

שאלה 2

Form

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

namespace targil4_22
{
    public partial class lsbSelectedItem : Form
    {
        private List theList = new List();
        string basisName = "";
        public lsbSelectedItem()
        {
            InitializeComponent();
        }


        private void btnOrderNew_Click(object sender, EventArgs e)
        {
            btnExit.Visible = false;
            btnOrderNew.Visible = false;
            lbBase.Visible = true;
            lstSelected.Visible = true;
            pnlBase.Visible = true;
            rdbEgg.Visible = true;
            rdbRice.Visible = true;
            btnAddBase.Visible = true;
            lblContain.Visible = true;
            picBox.Visible = false;

        }

        private void btnAddBase_Click(object sender, EventArgs e)
        {
            btnAddBase.Enabled = false;
            btnAddExtra.Visible = true;
            clbExtra.Visible = true;
            lblChooseExtra.Visible = true;
            btnremove.Visible = true;

            basis baseDish = new basis(basisName);
            if (theList.AddItem(baseDish))
            {
                theList.sum += baseDish.price;
                MessageBox.Show("The item added");
            }
            lstSelected.Items.Add(baseDish.GetName());

        }

        private void rdbRice_Click(object sender, EventArgs e)
```

```
        {
            basisName = rdbRice.Text;
            btnAddBase.Enabled = true;
        }

        private void rdbEgg_Click(object sender, EventArgs e)
        {
            basisName = rdbEgg.Text;
            btnAddBase.Enabled = true;
        }

        private void btnAddExtra_Click(object sender, EventArgs e)
        {

            if (clbExtra.CheckedItems.Count <= 5 &&
clbExtra.CheckedItems.Count > 0)
            {

                for (int i = 0; i < clbExtra.CheckedItems.Count; i++)
                {
                    extra extradish = new
extra(clbExtra.CheckedItems[i].ToString());
                    theList.sum += extradish.price;

                    if (theList.AddItem(extradish))
                    {
                        theList.allItems[i] = extradish;
                    }

                    lstSelected.Items.Add(extradish.GetName());

                }

                MessageBox.Show("The " + clbExtra.CheckedItems.Count + "
item added");

                btnremove.Enabled = true;
                btnSend.Visible = true;


            }
            else
            {
                MessageBox.Show("Please selected 1-5 items");
            }
        }

        private void clbExtra_ItemCheck(object sender, ItemCheckEventArgs
e)
        {
            btnAddExtra.Enabled = true;
        }

        private void btnremove_Click(object sender, EventArgs e)
        {
            if (lstSelected.Items.Count == 1)
            {
                MessageBox.Show("The list must contain at least one
product!");
                btnAddExtra.Enabled = false;
                return;

            }
```

```
            int ind = lstSelected.SelectedIndex;
            if (theList.allItems[ind] is extra)
            {
                theList.sum -= (theList.allItems[ind] as extra).price;
            }
            else
            {
                theList.sum -= (theList.allItems[ind] as basis).price;
            }

            theList.RemoveItem(ind - 1);

            lstSelected.Items.RemoveAt(ind);

            for (int i = 0; i < clbExtra.Items.Count; i++)
            {
                if (theList.allItems[ind - 1].GetName() ==
clbExtra.Items[i].ToString())
                {
                    clbExtra.SetItemChecked(i, false);
                    break;
                }

            }

        }

        private void btnSend_Click(object sender, EventArgs e)
        {
            timerSend.Start();
            btnremove.Visible = false;
            txtPrice.Text = theList.ToString();

        }


        private void timerSend_Tick(object sender, EventArgs e)
        {
            timerSend.Stop();
            grbBox.Visible = false;
            btnSend.Visible = false;

            MessageBox.Show("forwarded to reception");
            txtPrice.Visible = true;
            lblPriceTotal.Visible = true;
        }

        private void btnExit_Click(object sender, EventArgs e)
        {
            Close();
        }
    }
}
```

List

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.CompilerServices;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace targil4_22
{
    internal class List
    {
        public Item[] allItems;
        private int numOfItems;
        public int sum = 0;

        public List()
        {
            allItems = new Item[6];
            numOfItems = 0;
        }

        public Item[] GetItems() { return allItems; }////מחזירה מערך שלם
מוצרים

        public bool CanAddAnotherItems() { return numOfItems <
 allItems.Length; }//בודקת את ההוספה לרשימה


        public bool AddItem(Item I)
        {
            if (CanAddAnotherItems())
            {

                allItems[numOfItems] = I;
                numOfItems++;
                return true;
            }

            return false;
        }
        public void RemoveItem(int ind)
        {

            for (int j = ind; j < numOfItems - 1; j++)
            {
                allItems[j] = allItems[j + 1];
            }
            allItems[numOfItems - 1] = null;
            numOfItems--;////הקטנת את מספר המוצרים ברשימה שלה במוצרים

        }

        public override string ToString()
        {
            string price = sum.ToString();
            return price;
        }
```

```
    }
}
```

basis

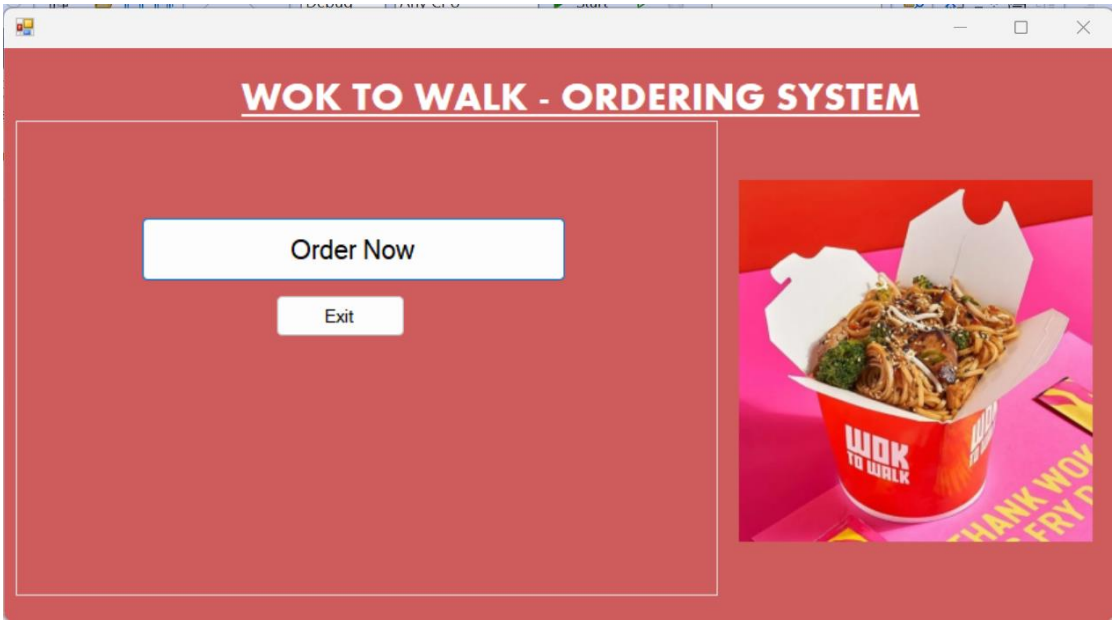
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace targil4_22
{
    internal class basis : Item
    {
        public int price = 10;


        public basis(string name) : base(name)
        {

        }


    }
}
```

extra

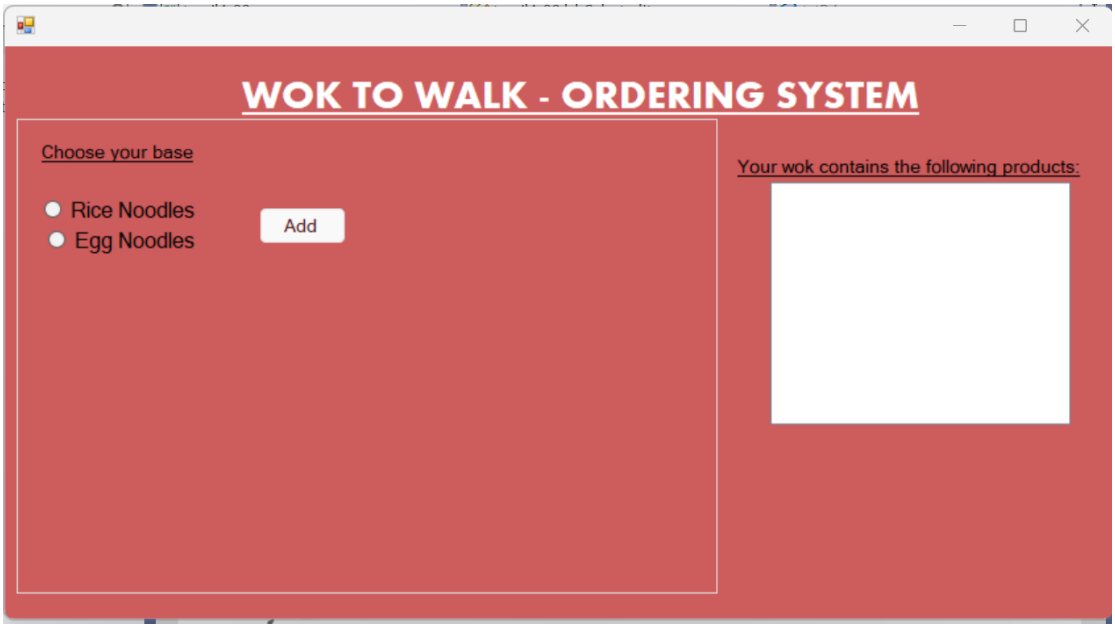
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace targil4_22
{
    internal class extra : Item
    {
        public int price = 5;


        public extra(string name) : base(name)
        {

        }

    }
}
```

פלטים –

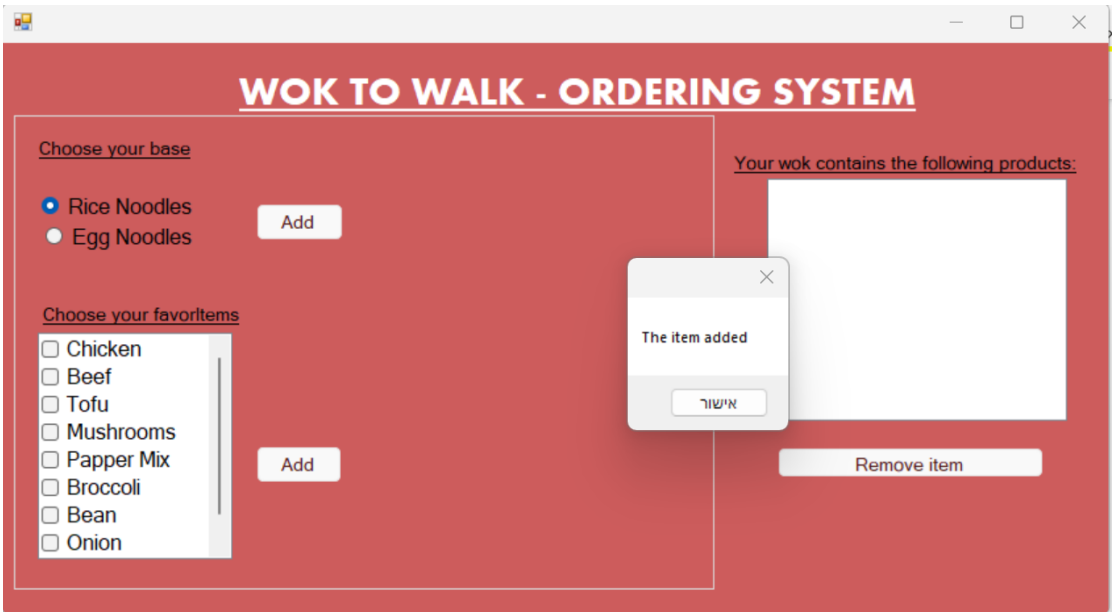
חלון ראשון – הזמנת אוכל סיני (מוקפץ) אופציה להתחלת הזמנה או ליציאה מהתוכנית.



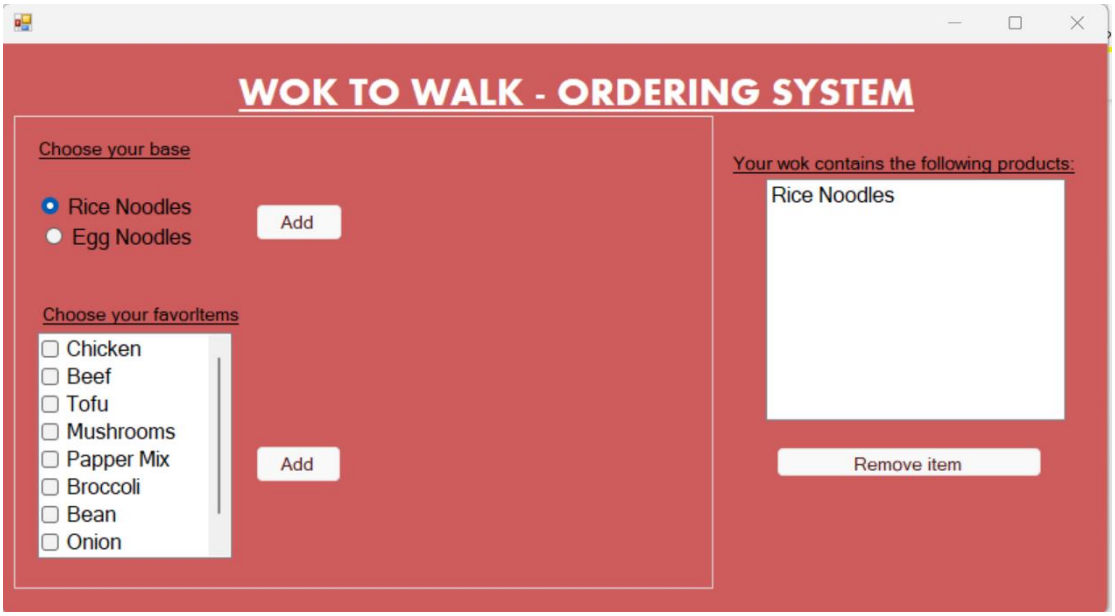
חלון שני – לאחר לחיצה על כפתור ההזמנה נפתח חלון של בחירת בסיס המוקפץ (סוג נודלס, מחיר 10 שקלים).
ניתן לבחור רק אופציה אחת ובמידה ולא לוחצים לא ניתן להתקדם בהזמנה, מימין יהיה ניתן לראות את הפרטים העתידים של ההזמנה .



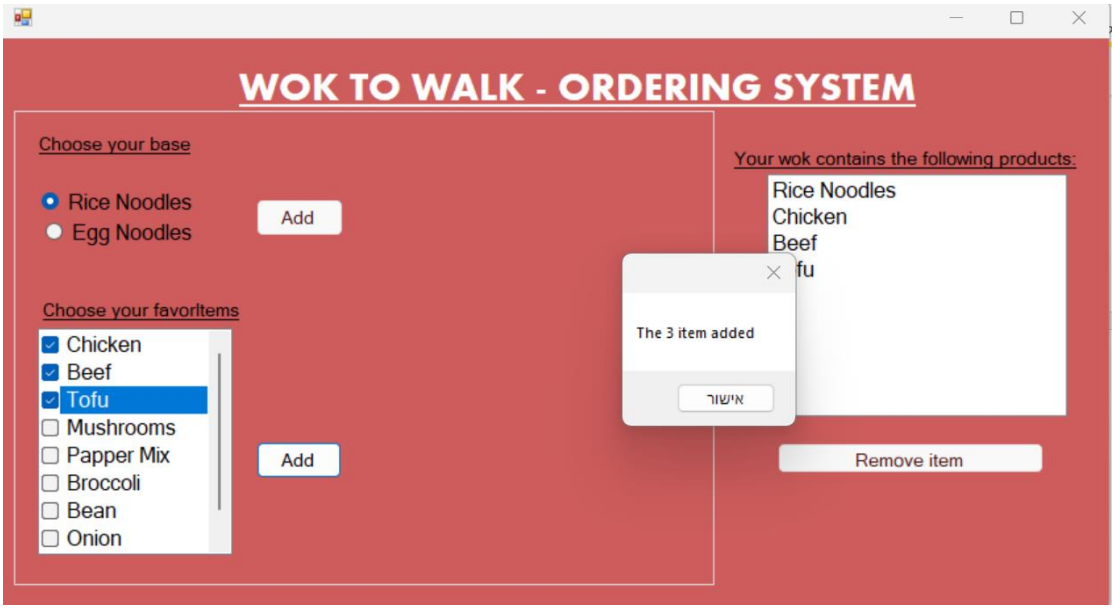
לאחר בחירת הבסיס הרצוי ישנו אישור של הוספת סוג הנודלס המועדף והצגתו בפרטי ההזמנה.



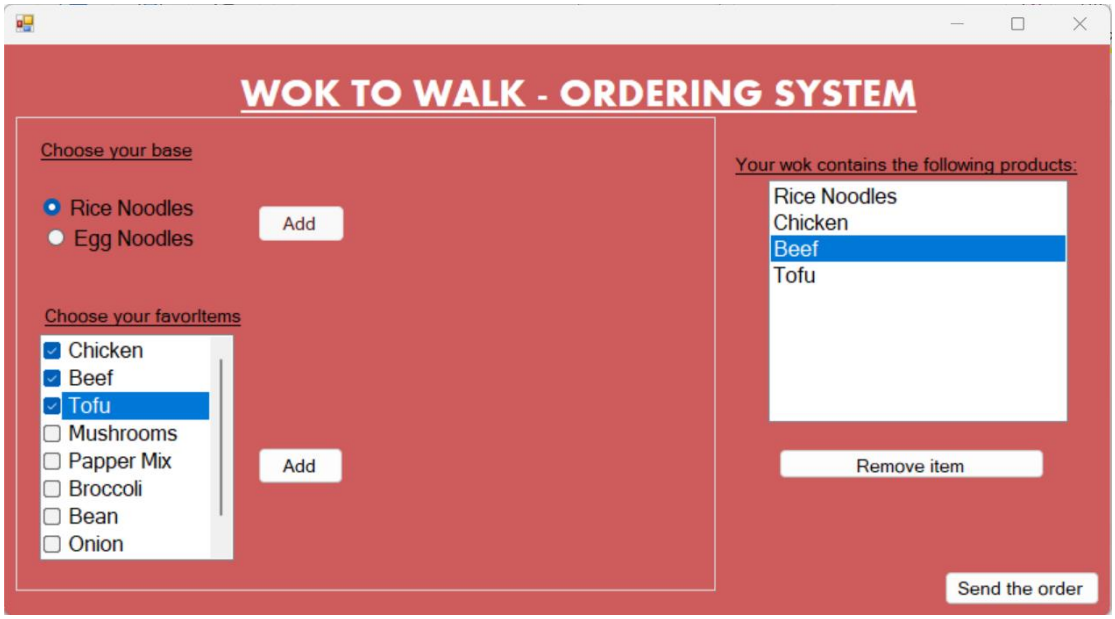
לאחר האישור, נפתחה אופציה של הוספת תוספות למוקפץ לפי בחירת הלקוח (כל תוספת עלות של 5 שקלים)



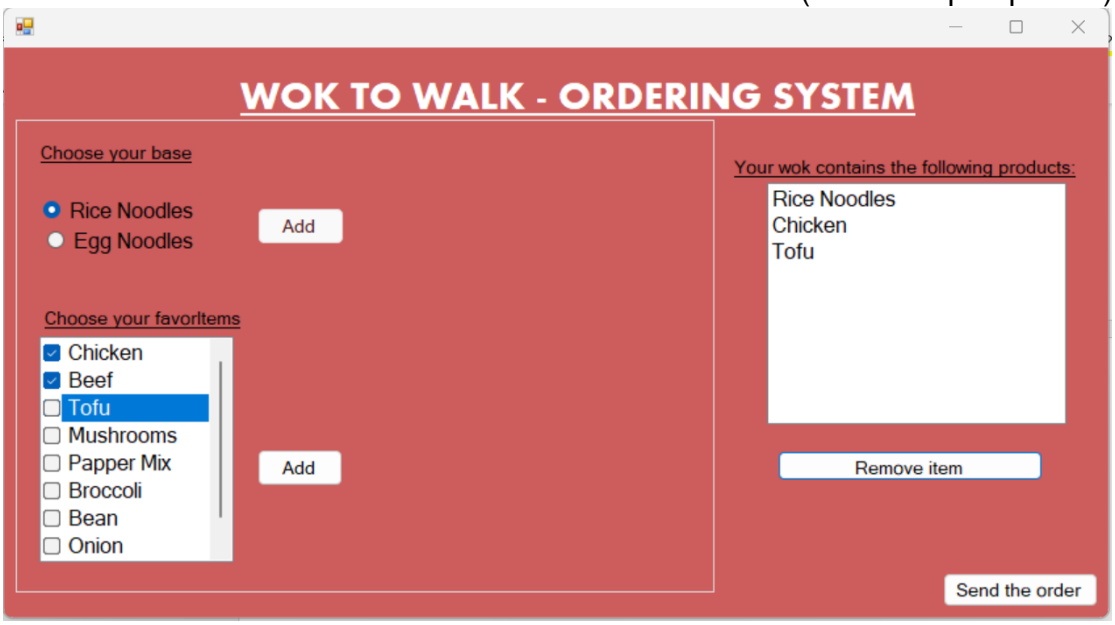
בחירת פריטים רצויים ואישור הוספה.



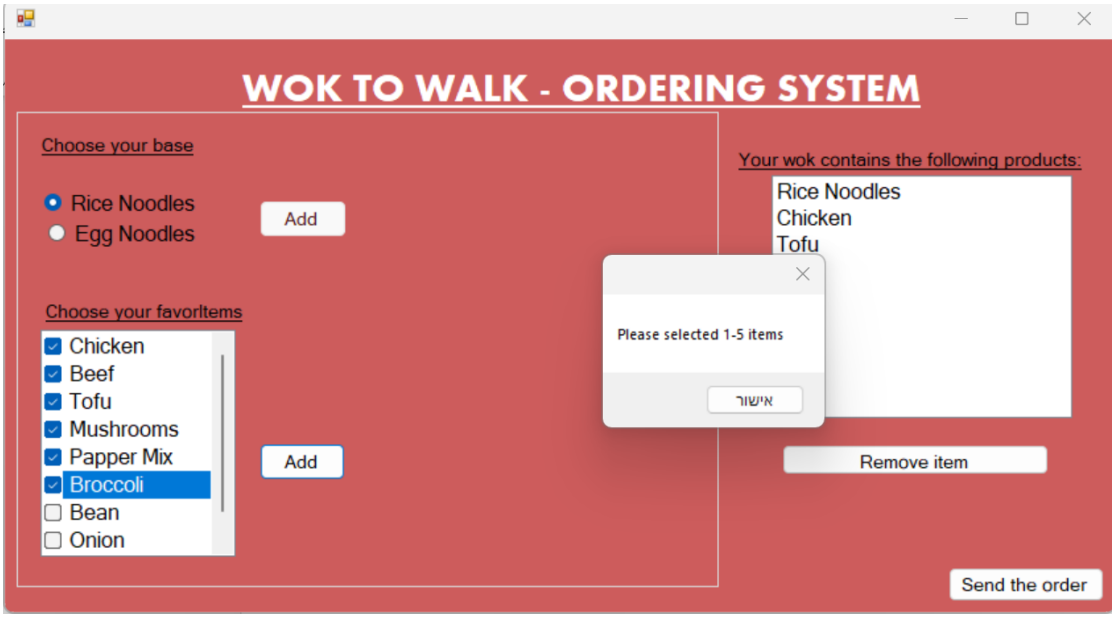
לאחר אישור הוספת המוצרים נפתחה אופציה של מחיקת מוצר או שליחת ההזמנה להכנה +תשלום.
ניתן להוסיף עד 6 מוצרים בפעם אחת.



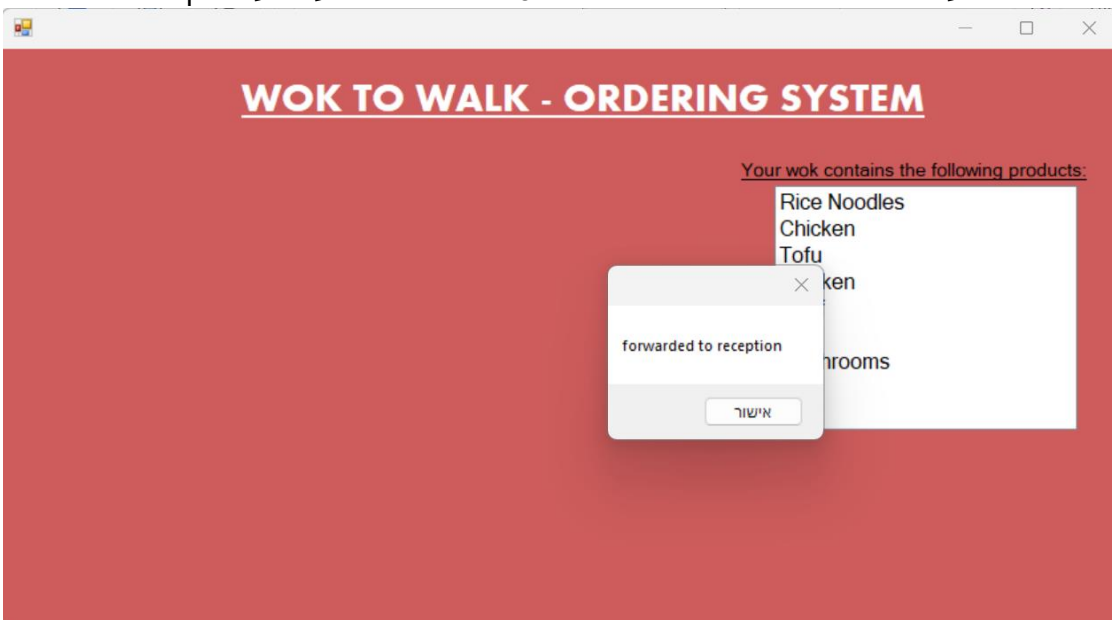
במידה והלקוח מתחרט על אחת מהתוספת (לדוגמה בשר) ניתן למחוק והקליק נעלם מהרשימה (מצד ימין=ניתן לבחור שוב).



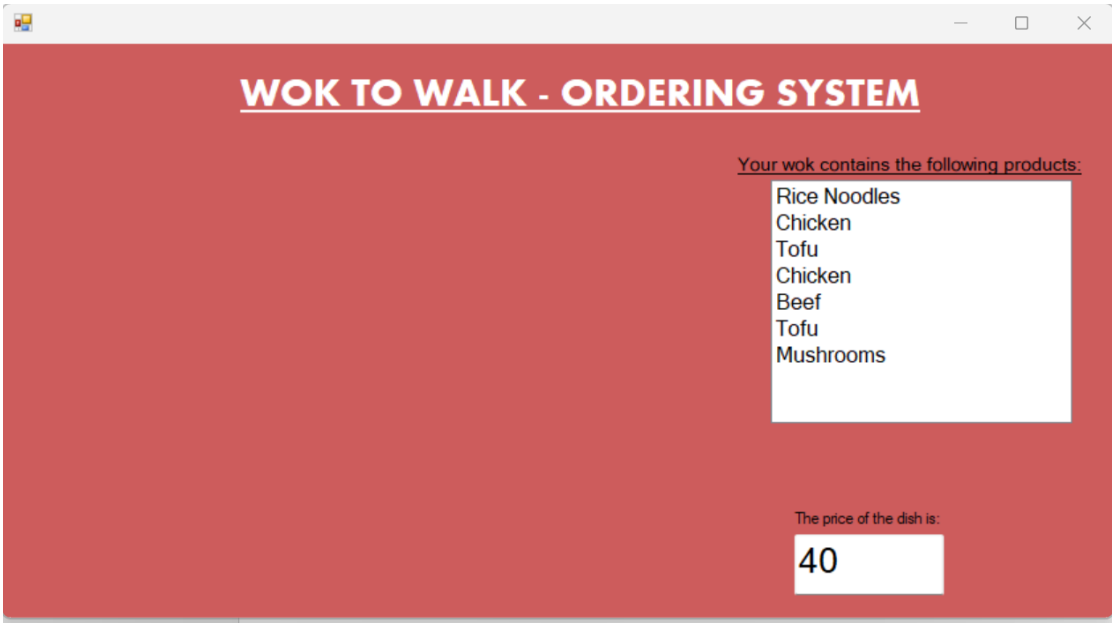
הודעת שגיאה שלא ניתן להוסיף יותר מ5 פריטים לתוך הנודלס



לאחר לחיצה על שליחת ההזמנה ישנו טיימר הסופר 3 שניות ומציג הודעת מעבר לקבלה.



לאחר 3 שניות עברנו למסך הסופי בו מוצגת ההזמנה במלואה ומחיר המנה לפי בחירת הלקוח .



בתאבון ☺