מטלה 2 GUI

מגישות:

אלינור וייצמן 321737165

הילור אלל 206396657

הנחות יסוד:

בכל מחלקה יש הערות המסבירות את רוב שורות הקוד והפעולות שבוצעו,
חלק מההערות בעברית מהמטלה הקודמת, ההערות שונו לאנגלית מטעמי נוחות.
כל מחלקה רשומה בדף נפרד לנוחות הקורא.

שאלה 1:

Form1:

```
using Assign4_1;
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;

namespace Assign4_1
{
    public partial class Form1 : Form
    {
        // Define Queue of Data Files
        QueueFiles queue;
        public Form1()
        {
            InitializeComponent();

            // Set title of the window
            this.Text = "My Queue Files";
        }

        // Helper function to hise the Menu and display new Group Box
        private void HideMenuDisplayNewGrp(GroupBox groupBox)
        {
            grpMenuOp.Visible = false; // Hide Menu
            groupBox.Visible = true; // Display new Group Box
        }

        // Helper function to reset app - return to the menu
        private void ResetMenu()
        {
            grpMenuOp.Visible = true; // Display Menu section
            lstMenuOp.ClearSelected(); // Reset selection in the list
            btnMenuOp.Enabled = false; // Disable the button "Submit"
        }
```

```
        private void Form1_Load(object sender, EventArgs e)
        {
            btnMenuOp.Enabled = false;


            // First Message
            MessageBox.Show("There are no files in queue yet",
"Greeting");

            // Create Queue of Data Files
            queue = new QueueFiles();
        }

        private void lstMenuOp_SelectedIndexChanged(object sender,
EventArgs e)
        {
            btnMenuOp.Enabled = true; // Enable Submit button
        }

        private void btnMenuOp_Click(object sender, EventArgs e)
        {
            DataFile dataFile; // Temp DataFile object to operate the
options from the Menu

            int option = (int)lstMenuOp.SelectedIndex; // Get the index of
the choice from ListBox lstMenuOp

            switch (option)
            {
                case 0: /* Create Default File */
                    dataFile = new DataFile(); // Activate default ctor
                    queue.Enqueue(dataFile); // Add to queue
                    MessageBox.Show("New Default File was created :)",
"New default file");

                    ResetMenu(); // Return and reset the Menu
                    break;

                case 1: /* Create New Data File */
                    HideMenuDisplayNewGrp(grpOp2); // Hide Menu and
display grpOp2
                    break;

                case 2: /* Remove file from queue (FIFO) */
                    dataFile = queue.Dequeue(); // Remove the first file
in the queue (FIFO)

                    // Check if Dequeue operation was successful
                    if (dataFile != null)
                        MessageBox.Show("File removed from queue:\n\n" +
dataFile.Dir(), "Removed file");
                    else
                        MessageBox.Show("The queue is empty :(", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);

                    ResetMenu(); // Return and reset the Menu
                    break;

                case 3: /* Print all files in queue */
                    if (queue.IsEmpty())
                        MessageBox.Show("The queue is empty :(", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
                    else
                        MessageBox.Show(queue.PrintQueue(), "Print all
files");

                    ResetMenu(); // Return and reset the Menu
                    break;

                case 4: /* Serach file by type - array of the same type */
                    if (queue.IsEmpty())
                    {
                        MessageBox.Show("The queue is empty :(", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                        ResetMenu(); // Return and reset the Menu
                    }
                    else
                        HideMenuDisplayNewGrp(grpOp5);

                    break;

                case 5: /* Biggest file in queue */
                    dataFile = queue.BigFile(); // Returns the biggest
file in queue

                    // Check If the Search was successful and show the
result
                    if (dataFile != null)
                        MessageBox.Show($"The biggest file in queue:\n\n
{dataFile.Dir()}", "Biggest file");
                    else
                        MessageBox.Show("The queue is empty :(", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);

                    ResetMenu(); // Return and reset the Menu
                    break;

                case 6: /* Exit the app */
                    MessageBox.Show("Bye Bye :)", "Good Bye");
                    this.Close(); // Close the GUI window
                    break;

                default:
                    break;
            }

        }

        private void btnNewFile_Click(object sender, EventArgs e)
        {
            /* Validation of the fields from grpOp2 */
            if (txtNewFileName.Text.Trim().Equals("")) // File Name is
empty
                MessageBox.Show("Must enter File Name :(", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);

            else if (cmbBoxNewFileType.SelectedItem == null) // User didnt
choose a type
                MessageBox.Show("Must Select File Type :(", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);

            else // Valid inputs
            {
                // Get file name
                string fileName = txtNewFileName.Text;
```

```
                // Get file data
                string fileData = txtBoxNewFileData.Text;
                // Get file type and cast the choice (from ComboBox) into
enum
                int indexFromCmbBox = cmbBoxNewFileType.SelectedIndex;
                indexFromCmbBox++;
                FileTypeExtention fileType =
(FileTypeExtention)(indexFromCmbBox);
                // Create Data File with argumented contructor
                DataFile dataFile = new DataFile(fileName, fileData,
fileType);

                // Add to queue
                if (queue.Enqueue(dataFile)) // Success
                {
                    txtNewFileName.Clear(); // Clear TextBox
                    txtBoxNewFileData.Clear(); // Clear TextBox
                    cmbBoxNewFileType.SelectedItem = null; // Clear
ComboBox

                    MessageBox.Show("New Data File was created :)", "New
data file");
                    grpOp2.Visible = false; // Hide GroupBox grpOp2
                    ResetMenu(); // Return and reset the Menu
                }
                else // Failure
                {
                    txtNewFileName.Clear(); // Clear file name TextBox
                    txtBoxNewFileData.Clear(); // Clear file data TextBox
                    cmbBoxNewFileType.SelectedItem = null; // Clear file
type ComboBox

                    MessageBox.Show("The file is already exist in queue
:(", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
        }

        private void btnSearchType_Click(object sender, EventArgs e)
        {

            if (cmbBoxSelectType.SelectedItem == null)
                MessageBox.Show("Must Select File Type :(", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);

            else // Search for files with selected type
            {
                // Cast selected item to FileTypeExtention enum

                int indexFromCmbBox = cmbBoxSelectType.SelectedIndex;
                indexFromCmbBox++;
                FileTypeExtention fileTypeToSearch =
(FileTypeExtention)(indexFromCmbBox);

                DataFile[] arrFilesSelectedType =
queue.SearchFileByType(fileTypeToSearch);

                if (arrFilesSelectedType == null) // Failure
                {
                    cmbBoxSelectType.SelectedItem = null; // Clear
ComboBox
                    MessageBox.Show("No files found from the selected type
:(", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
                }

                else // Success
                {
                    string resMsgBox = "The files from type:\n"; // Output
string

                    // Iterate through the array (arrFileSelectedType)
                    foreach (var file in arrFilesSelectedType)
                        resMsgBox += file.Dir();

                    cmbBoxSelectType.SelectedItem = null; // Clear
ComboBox

                    MessageBox.Show(resMsgBox, "Files by type"); // Print
the files details to MessageBox
                    grpOp5.Visible = false; // Hide the GroupBox grpOp5
                    ResetMenu(); // Return and reset the Menu
                }
            }
        }

    }
}
```

QueueFiles:

```
using Microsoft.SqlServer.Server;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Policy;
using System.Text;
using System.Threading.Tasks;

namespace Assign4_1
{
    internal class QueueFiles
    {
        DataFile[] arr;
        int index; //מציין את המקום הפנוי בתור

        //constructors
        //בנאי ריק
        public QueueFiles()
        {
            arr = new DataFile[0];
            index = -1;
        }

        //מתודה הבודקת אם התור ריק
        public bool IsEmpty()
        {
            if (index == -1)
            {
                return true;
            }
            return false;
        }

        //מתודה המוסיפה קובץ לסוף המערך
        public bool Enqueue(DataFile file)
        {
            for (int i = 0; i < index; i++)
                if (CompareFiles.EqualFiles(arr[i], file))
                    return false;

            if(index==-1) //מקרה קצה הוספה לתור ריק
            {
                this.arr = new DataFile[1];
                arr[0] = file;
                index = 1;
            }
            else
            {
                Array.Resize(ref arr, arr.Length+1); // הגדלת גודל המערך ועדכון
ההפנייה להפנייה החדשה
                index++; //קידום האינדקס
                arr[index-1] = file; //הוספה לסוף התור
            }

            return true;
        }

        //מתודה המוציאה קובץ מהתור ומחזירה הפניה לקובץ היוצא
        public DataFile Dequeue()
        {
```

```
            if (IsEmpty())
            {
                Console.WriteLine("The queue is empty");
                return null;
            }
            if (index == 1)
            {
                DataFile file = arr[0];
                arr[0] = null;
                index = -1;
                return file;
            }
            else
            {
                DataFile file = arr[0]; // שמירה במשתנה עזר את ההפנייה לאיבר במקום
0ה
                DataFile[] arrtemp = new DataFile[index-1];
                int j = 0;
                for (int i = 1; i < index; i++)
                {
                    arrtemp[j++] = arr[i]; //צמצום האיברים
                }
                index--;//הקטנת האינדקס
                arr=arrtemp;
                return file;
            }
        }
        //מתודה המחזירה הפניה לקובץ שגודלו הגבוה ביותר
        public DataFile BigFile()
        {
            if (IsEmpty())
                return null;

            if (index == 1)
                return arr[0];

            DataFile dataFile;
            DataFile biggestFile = Dequeue();
            Enqueue(biggestFile);

            for (int i = 0; i < index-1; i++)
            {
                dataFile = Dequeue();
                if (CompareFiles.CompareSizeFiles(dataFile, biggestFile)
== 1)
                    biggestFile = dataFile;

                Enqueue(dataFile);
            }

            return biggestFile;
        }

        public string PrintQueue()
        {
            string res = "";

            if (IsEmpty())
            {
                return null;
            }

            DataFile tempFile;
```

```
            for (int i = 0; i < index; i++)
            {
                tempFile = Dequeue();

                res += tempFile.Dir() + "\n";

                Enqueue(tempFile);
            }
            return res;
        }

        private DataFile[] GetArr()
        {
            return arr;
        }
        public DataFile[] SearchFileByType(FileTypeExtention fileType)
        {
            if (!IsEmpty())
            {
                DataFile tempFile;
                QueueFiles typesQueue = new QueueFiles();

                for (int i=0; i<index;i++)
                {
                    tempFile = Dequeue();

                    if(tempFile.GetThisType()== fileType)
                        typesQueue.Enqueue(tempFile);

                    Enqueue(tempFile);
                }
                if (!typesQueue.IsEmpty())
                    return typesQueue.GetArr();
            }
            return null;
        }
    }
}
```

DataFile:

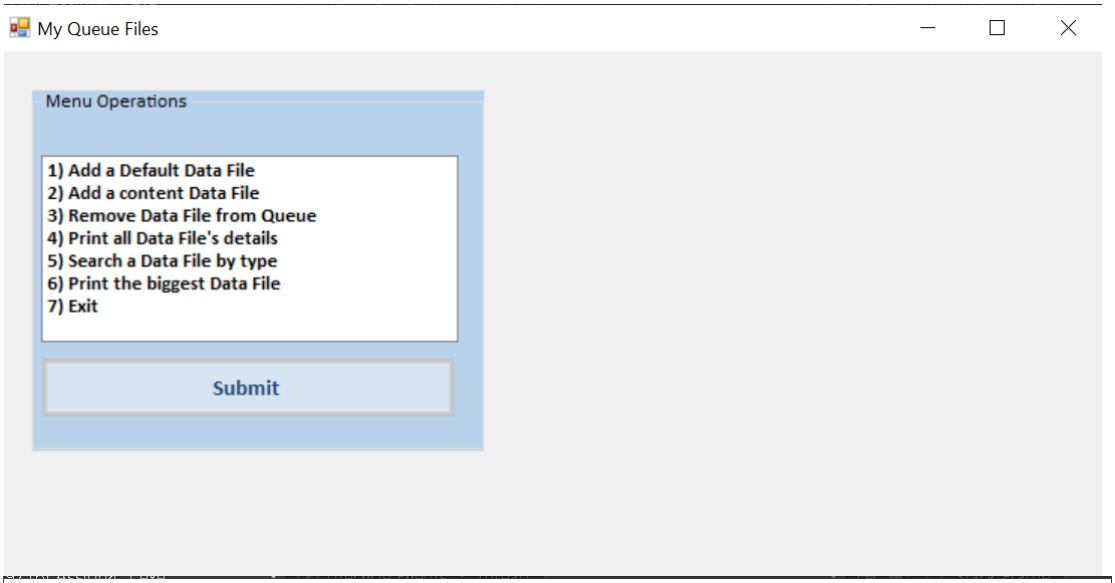
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Assign4_1
{
    public enum FileTypeExtention
    {
        TXT = 1, DOC, DOCX, PDF, PPTX
    }
    internal class DataFile
    {
        private string fileName;
        private DateTime lastUpdateTime =new DateTime();
        private string data;
        readonly FileTypeExtention type;
        static uint counter;
        static uint counterDefault;


        //constractors
        //בנאי המקבל פרמטרים:
        public DataFile(string fileName, string data, FileTypeExtention
type)
        {
            //while (SetFileName(fileName) == false)// אתחול שם הקובץ עד קבלת
ערך תקין
            //{
            //    Console.WriteLine("The file name is incorrect. Enter
file name:");
            //    fileName = Console.ReadLine();
            //}

            this.fileName = fileName;
            this.data = data;
            SetTime();//אתחול חתימת זמן לזמן הנוכחי
            this.type = type;

            counter++;
        }
        //בנאי ריק עם ערך ברירת מחדל:
        public DataFile() : this("SameFile" + (++counterDefault), "",
FileTypeExtention.TXT)
        {
        }

        public string GetData()
        {
            return data;
        }
        public void SetData(string data)
        {
            this.data = data;
        }

        public FileTypeExtention GetThisType()
        {
            return type;
```

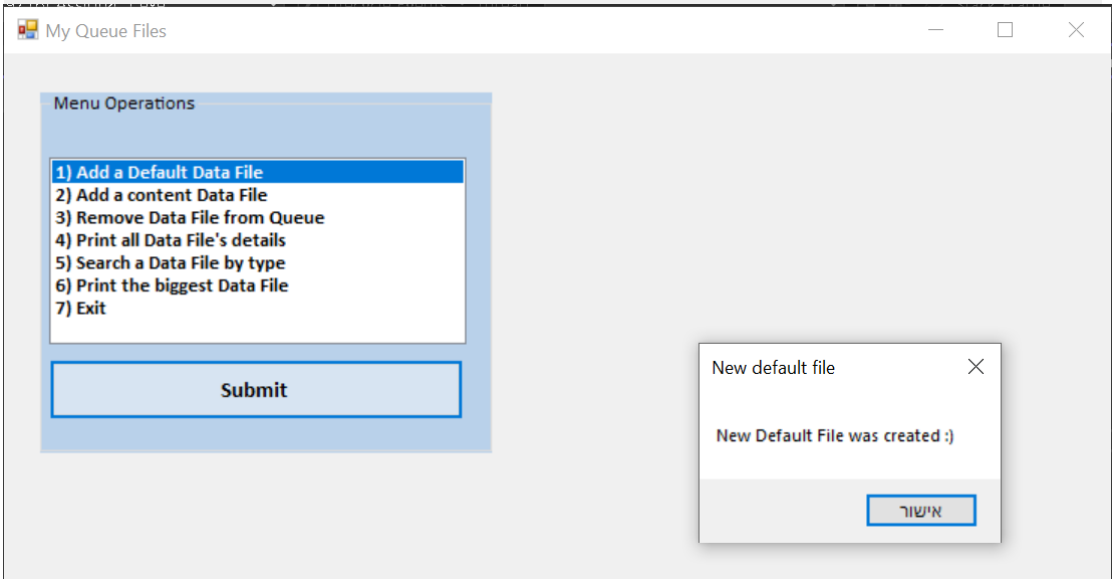
```
        }
        public string GetFileName()
        {
            return fileName;
        }
        //מתודה לשם הקובץ - בודקת שערכו תקין:
        public bool SetFileName(string fileName)
        {
            for (int i = 0; i < fileName.Length; i++)
            {
                if (fileName[i] == '\\' || fileName[i] == '/' ||
fileName[i] == ':' || fileName[i] == '*' || fileName[i] == '?' ||
fileName[i] == '<' || fileName[i] == '>' || fileName[i] == '|')
                {
                    Console.WriteLine("The file name can't contain any of
the following characters:\n \\ / : * ? \" < > |");
                    return false;
                }

            }
            this.fileName = fileName;
            return true;
        }
        //מתודות עבור חתימת זמן:
        private void SetTime()
        {
            lastUpdateTime = DateTime.Now; // current date/time based on
current system
        }
        public DateTime GetTime()
        {
            return lastUpdateTime;
        }

        //מתודה המחזירה את גודל הקובץ:
        public double GetSize ()
        {
            double size = GetData().Length;
            return size;
        }
        //מתודה המדפיסה את נתוני המחלקה:
        public string Dir()
        {
            return $"{lastUpdateTime} {GetSize() / 1024} KB
{fileName}.{type}";
        }
    }
}
```

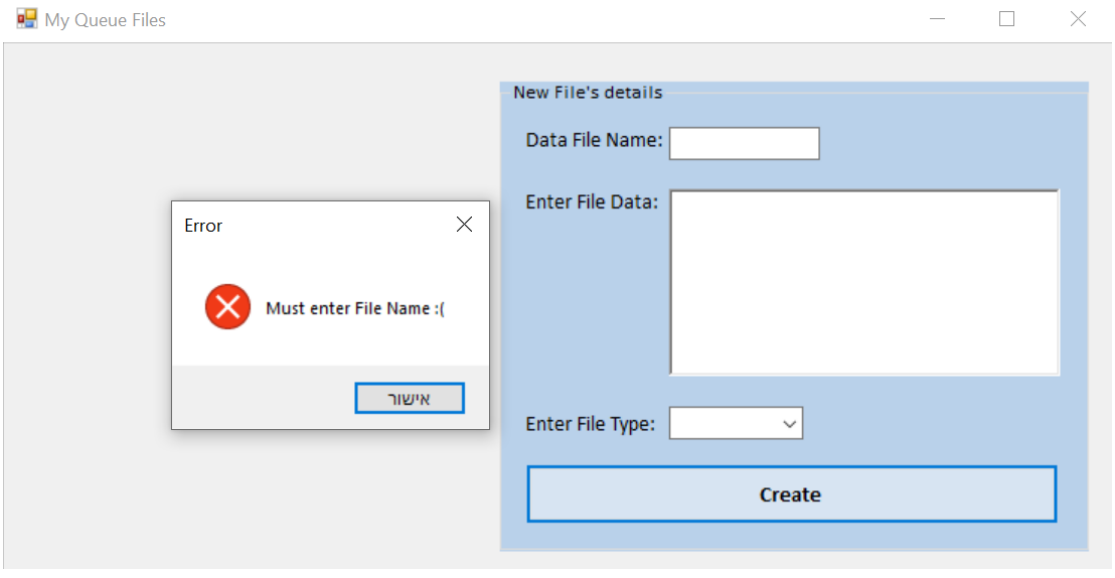
CompareFiles:

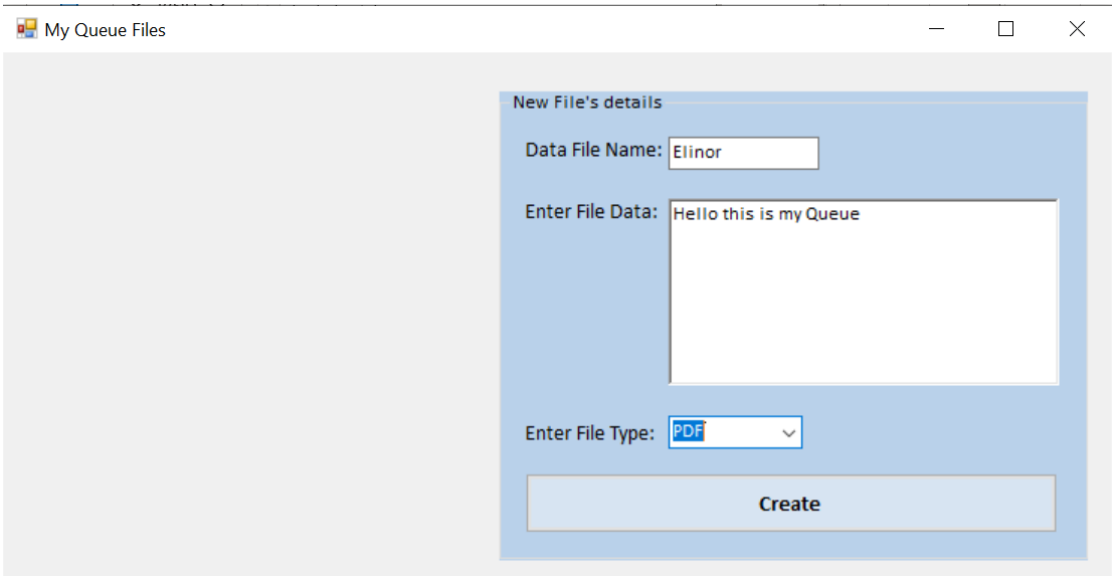
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Assign4_1
{
    internal static class CompareFiles
    {
        public static bool EqualFiles(DataFile file1, DataFile file2)
        {
            if (file1.GetFileName() == file2.GetFileName() &&
file1.GetData() == file2.GetData())
                return true;
            return false;
        }

        public static int CompareSizeFiles(DataFile file1, DataFile file2)
        {
            if (file1.GetSize() > file2.GetSize())
                return 1;
            if (file1.GetSize() < file2.GetSize())
                return -1;
            return 0;
        }
    }
}
```

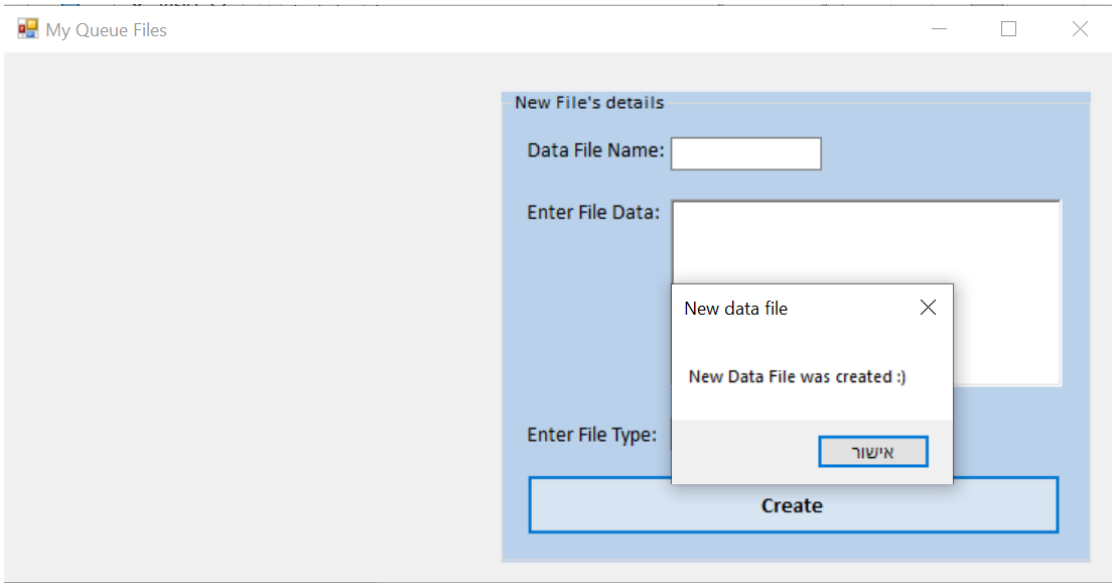
Outputs Q1:

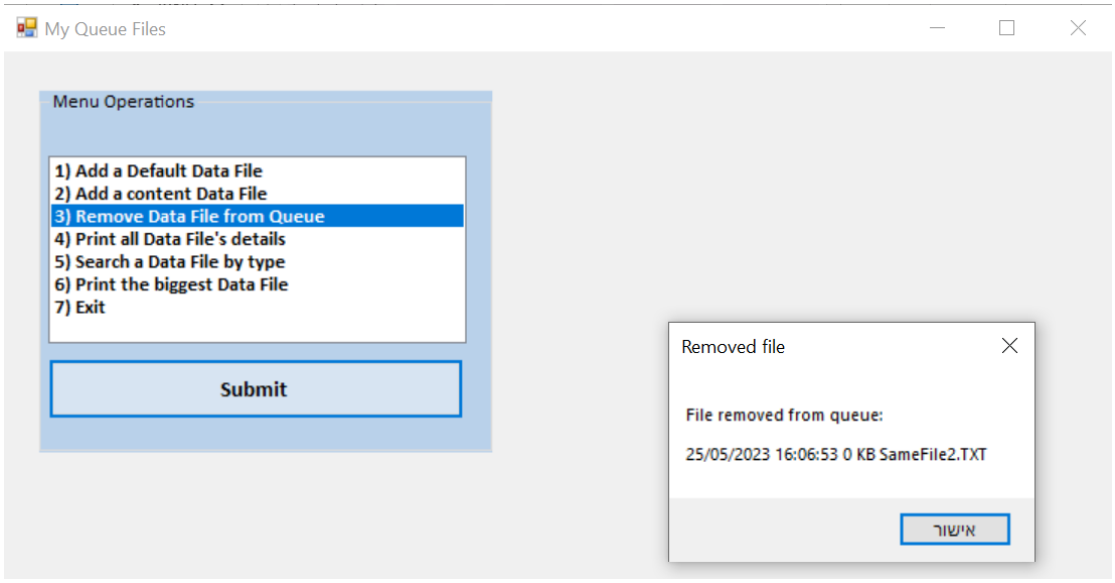

My Queue Files
Menu Operations
1) Add a Default Data File
2) Add a content Data File
3) Remove Data File from Queue
4) Print all Data File's details
5) Search a Data File by type
6) Print the biggest Data File
7) Exit
Submit

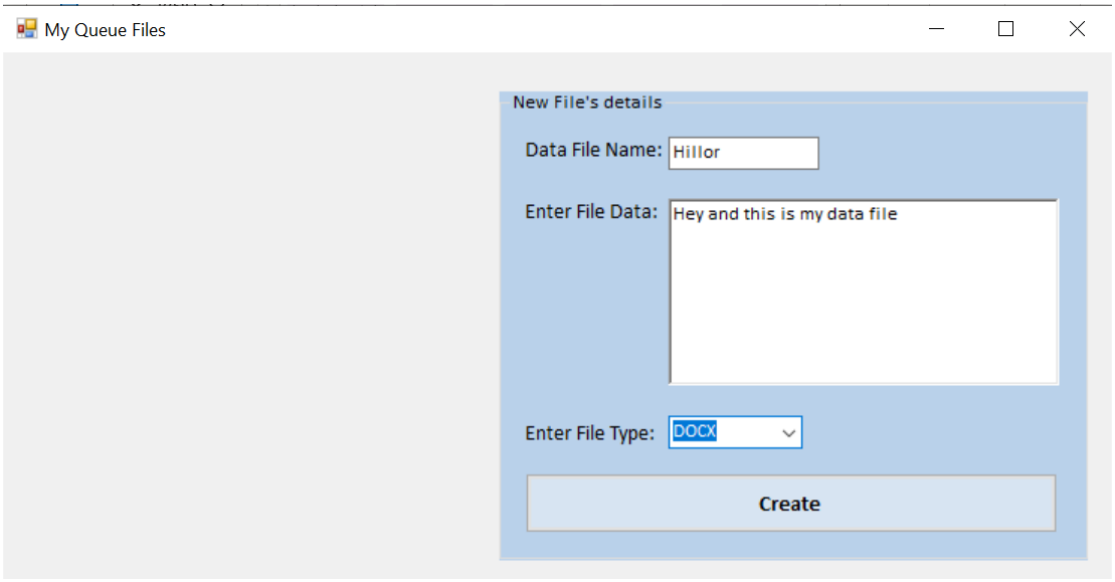

My Queue Files
Menu Operations
1) Add a Default Data File
2) Add a content Data File
3) Remove Data File from Queue
4) Print all Data File's details
5) Search a Data File by type
6) Print the biggest Data File
7) Exit
Submit
New default file
New Default File was created :)
אישור

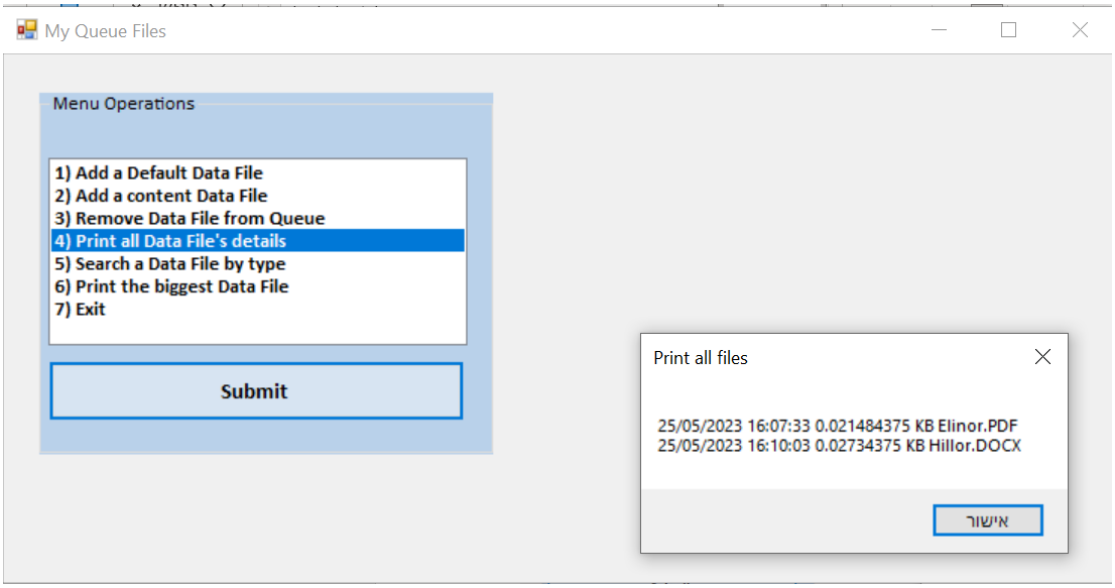

My Queue Files
New File's details
Data File Name:
Enter File Data:
Enter File Type:
Create
Error
Must enter File Name :(
אישור

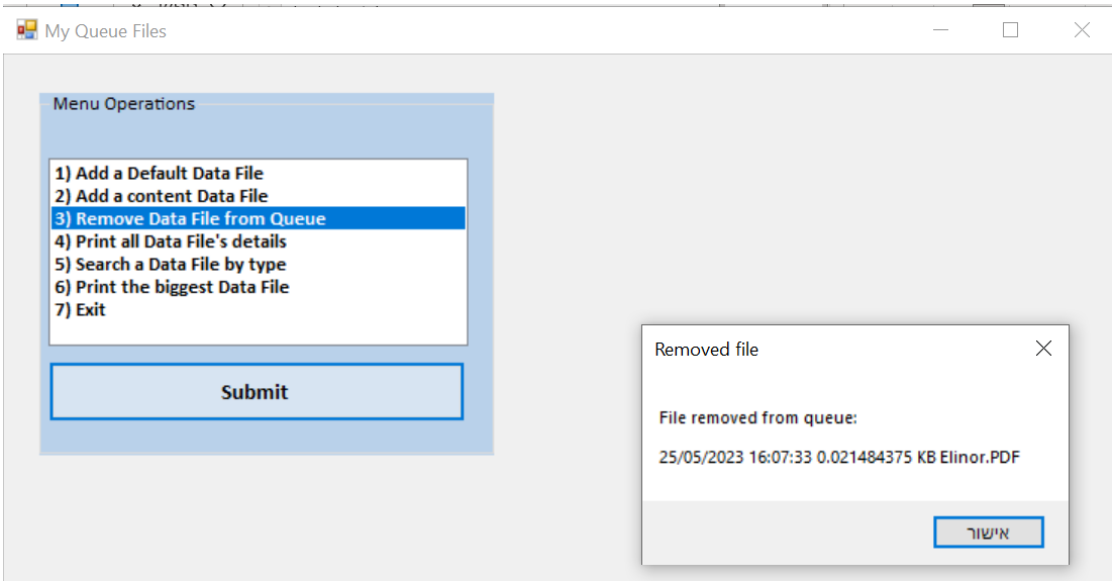

My Queue Files
New File's details
Data File Name:
Elinor
Enter File Data:
Hello this is my Queue
Enter File Type:
PDF
Create

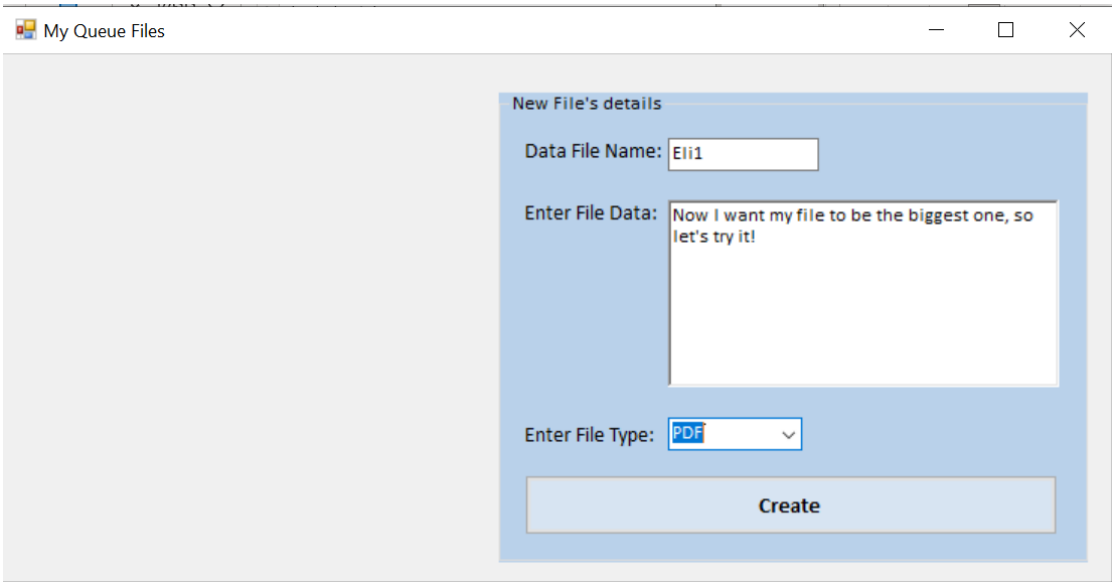

My Queue Files
New File's details
Data File Name:
Enter File Data:
New data file
New Data File was created :)
אישור
Enter File Type:
Create

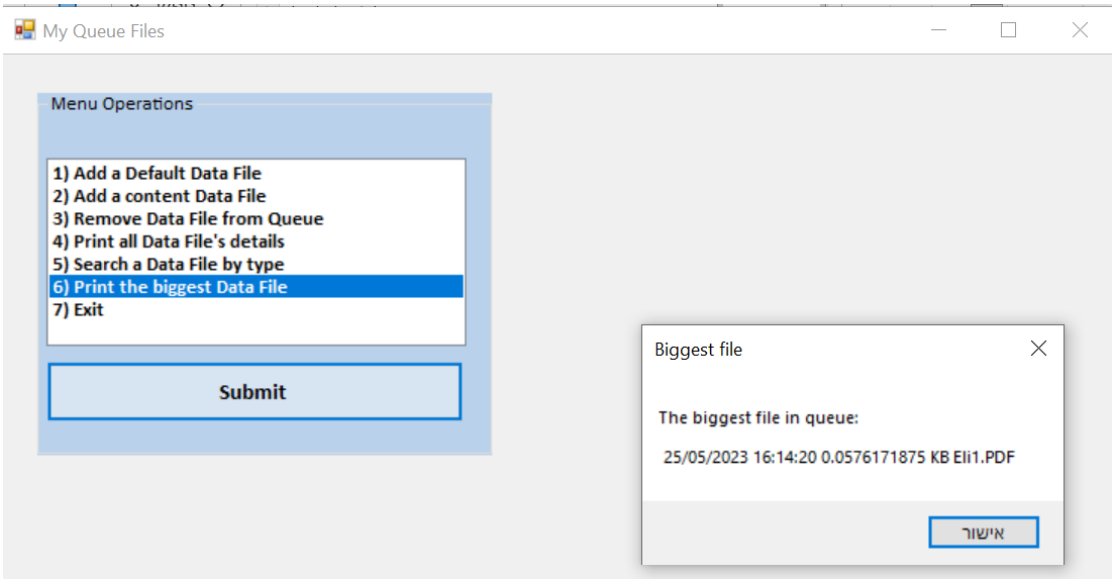

My Queue Files
Menu Operations
1) Add a Default Data File
2) Add a content Data File
3) Remove Data File from Queue
4) Print all Data File's details
5) Search a Data File by type
6) Print the biggest Data File
7) Exit
Submit
Removed file
File removed from queue:
25/05/2023 16:06:53 0 KB SameFile2.TXT
אישור

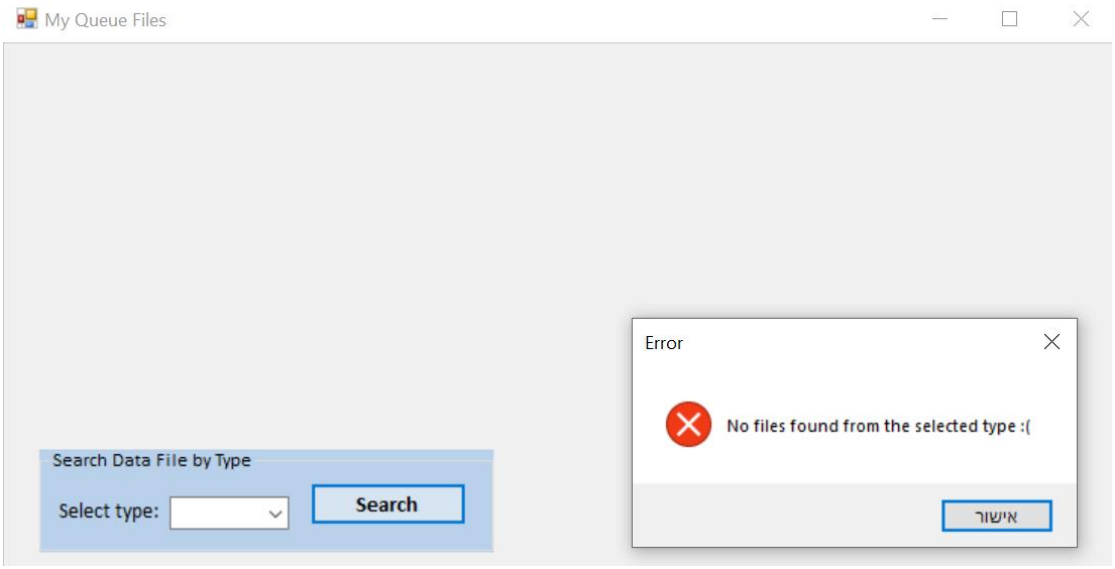

My Queue Files
New File's details
Data File Name:
Hillor
Enter File Data:
Hey and this is my data file
Enter File Type:
DOCX
Create

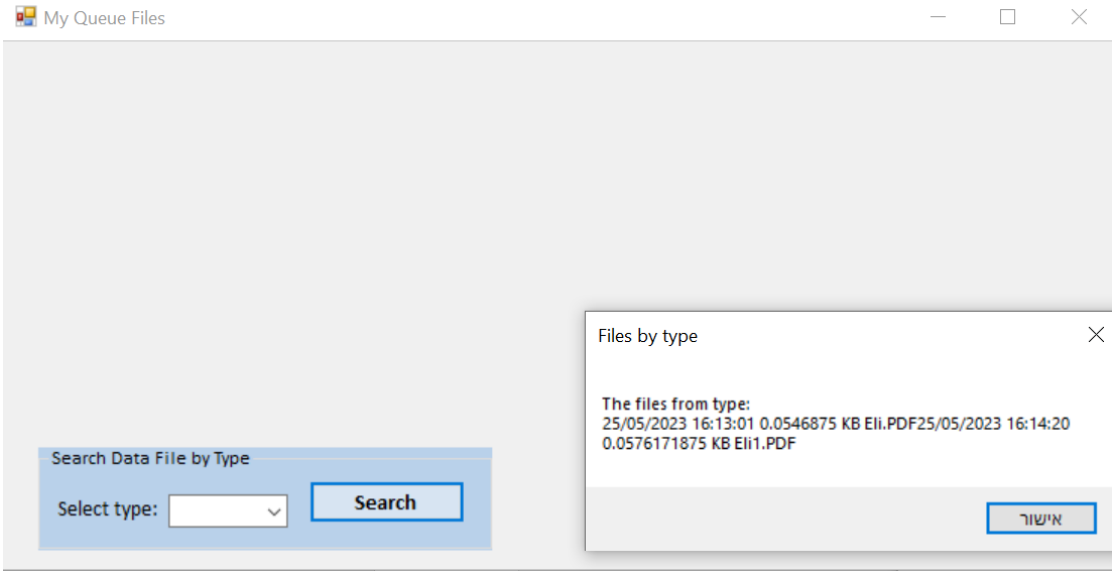

My Queue Files
Menu Operations
1) Add a Default Data File
2) Add a content Data File
3) Remove Data File from Queue
4) Print all Data File's details
5) Search a Data File by type
6) Print the biggest Data File
7) Exit
Submit
Print all files
25/05/2023 16:07:33 0.021484375 KB Elinor.PDF
25/05/2023 16:10:03 0.02734375 KB Hillor.DOCX
אישור

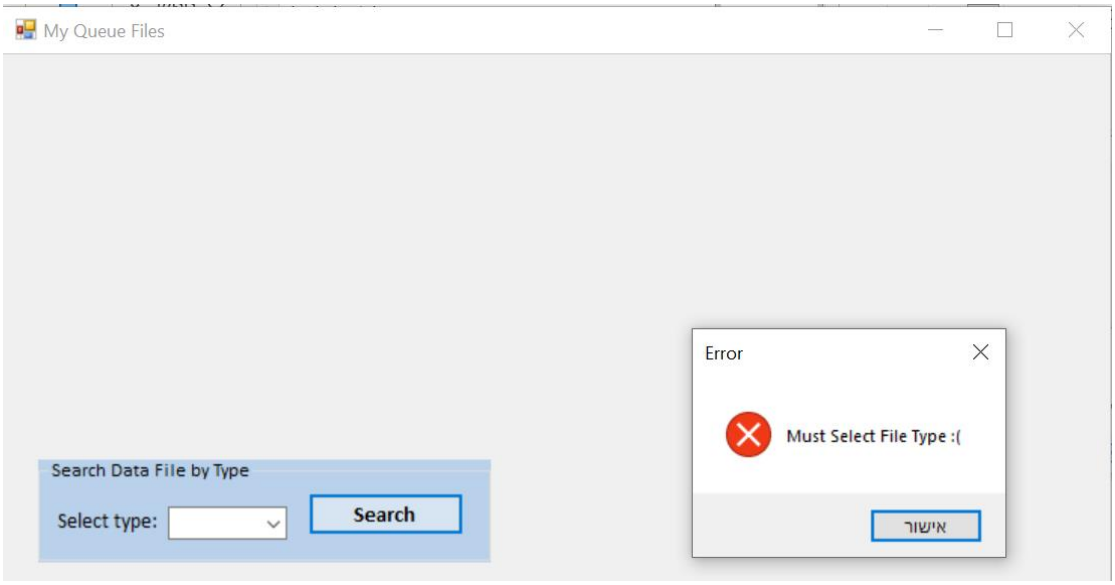

My Queue Files
Menu Operations
1) Add a Default Data File
2) Add a content Data File
3) Remove Data File from Queue
4) Print all Data File's details
5) Search a Data File by type
6) Print the biggest Data File
7) Exit
Submit
Removed file
File removed from queue:
25/05/2023 16:07:33 0.021484375 KB Elinor.PDF
אישור

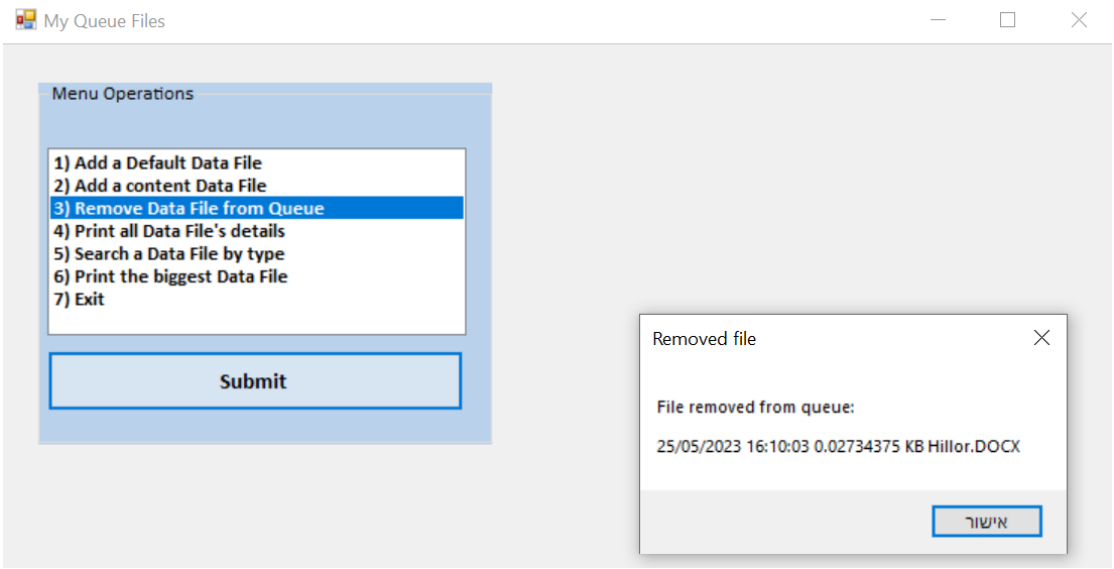

My Queue Files
New File's details
Data File Name:
Eli1
Enter File Data:
Now I want my file to be the biggest one, so let's try it!
Enter File Type:
PDF
Create

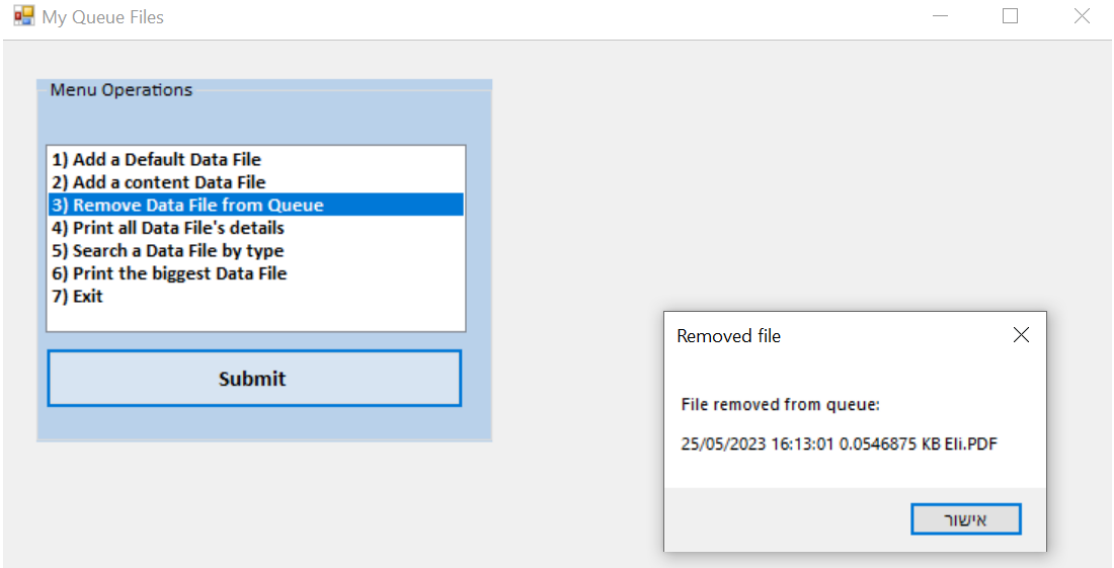

My Queue Files
Menu Operations
1) Add a Default Data File
2) Add a content Data File
3) Remove Data File from Queue
4) Print all Data File's details
5) Search a Data File by type
6) Print the biggest Data File
7) Exit
Submit
Biggest file
The biggest file in queue:
25/05/2023 16:14:20 0.0576171875 KB Eli1.PDF
אישור

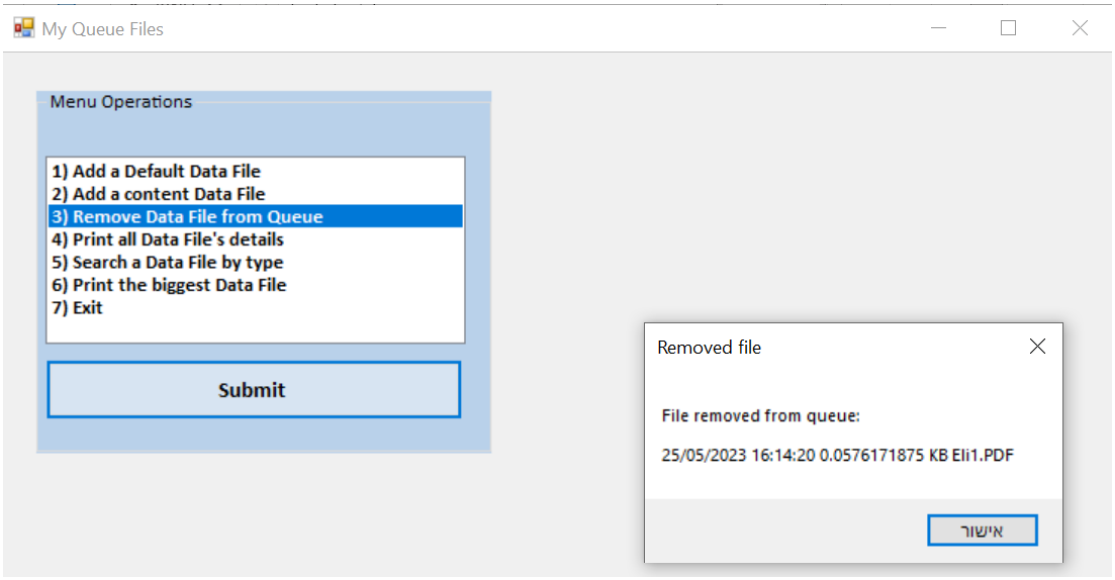

My Queue Files
Search Data File by Type
Select type:
Search
Error
No files found from the selected type :(
אישור

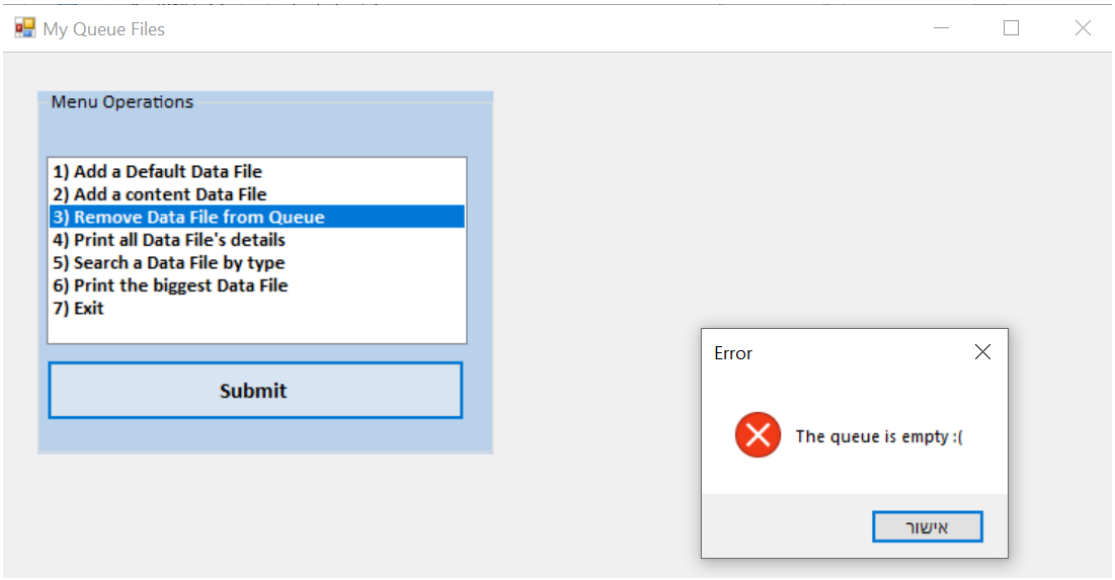

My Queue Files
Search Data File by Type
Select type:
Search
Files by type
The files from type:
25/05/2023 16:13:01 0.0546875 KB Eli.PDF25/05/2023 16:14:20
0.0576171875 KB Eli1.PDF
אישור

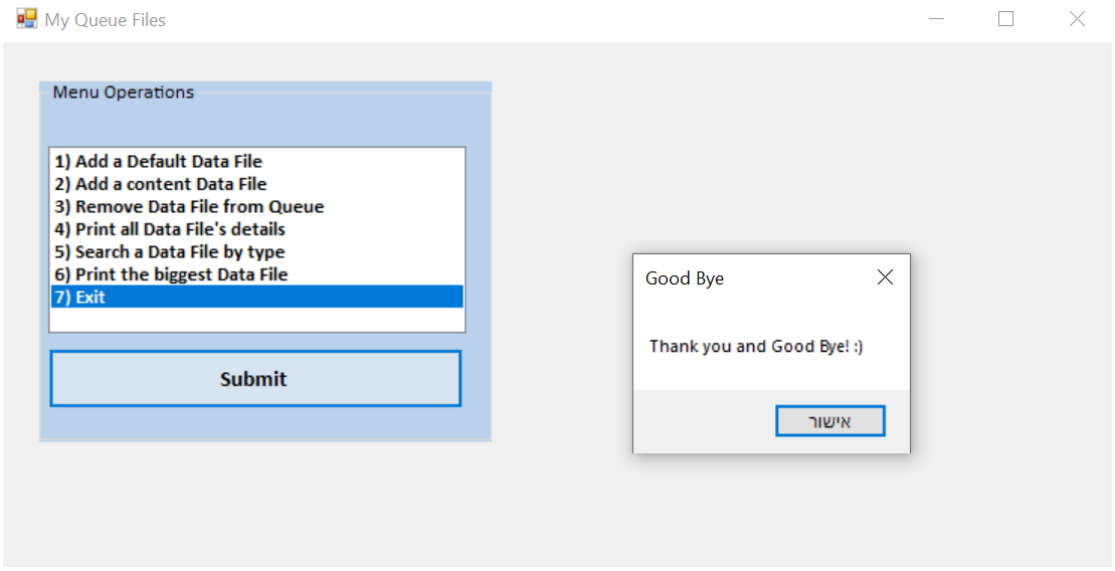

My Queue Files
Search Data File by Type
Select type:
Search
Error
Must Select File Type :(
אישור


My Queue Files
Menu Operations
1) Add a Default Data File
2) Add a content Data File
3) Remove Data File from Queue
4) Print all Data File's details
5) Search a Data File by type
6) Print the biggest Data File
7) Exit
Submit
Removed file
File removed from queue:
25/05/2023 16:10:03 0.02734375 KB Hillor.DOCX
אישור


My Queue Files
Menu Operations
1) Add a Default Data File
2) Add a content Data File
3) Remove Data File from Queue
4) Print all Data File's details
5) Search a Data File by type
6) Print the biggest Data File
7) Exit
Submit
Removed file
File removed from queue:
25/05/2023 16:13:01 0.0546875 KB Eli.PDF
אישור


My Queue Files
Menu Operations
1) Add a Default Data File
2) Add a content Data File
3) Remove Data File from Queue
4) Print all Data File's details
5) Search a Data File by type
6) Print the biggest Data File
7) Exit
Submit
Removed file
File removed from queue:
25/05/2023 16:14:20 0.0576171875 KB Eli1.PDF
אישור


My Queue Files
Menu Operations
1) Add a Default Data File
2) Add a content Data File
3) Remove Data File from Queue
4) Print all Data File's details
5) Search a Data File by type
6) Print the biggest Data File
7) Exit
Submit
Error
The queue is empty :(
אישור


My Queue Files
Menu Operations
1) Add a Default Data File
2) Add a content Data File
3) Remove Data File from Queue
4) Print all Data File's details
5) Search a Data File by type
6) Print the biggest Data File
7) Exit
Submit
Good Bye
Thank you and Good Bye! :)
אישור

<u>שאלה 2: הסבר על הממשק</u>

בנינו ממשק רכישה של מזון ברשת אשר נקראת "מק גוי".

הרשת בעלת לוגו וסלוגן משלה אשר באו לידי ביטוי בעיצוב התכנית.

הממשק מאפשר בחירת סוג מזון לרצון המשתמש.

מיד לאחר בחירת סוג, מופיע תפריט של מנות מאותו הסוג.

בתפריט מצוינים שם המנה, מחירה ליחידה אחת, ערך קלורי של המנה וסוגה.

לאחר הקלדת כמות המנות הרצויות, לנוחיות המשתמש ישנה אופציית חישוב

המחיר לכמות המנות שבחר.

לאחר הכנסת המנה לעגלה מתאפשרת כמובן הכנסה נוספת של מנות ו/או הורדת מנה מהעגלה בעת לחיצה עליה.

המשתמש יכול לרוקן את כלל העגלה ולהתחיל תהליך הזמנה מחדש.

נבדקו מקרי קצה בכלל החלונות ועל כן ברגע שהמשתמש מבצע פעולה לא נכונה מופיעה על המסך הודעה ידידותית המסבירה את הבעיה.

זמן ההזמנה מוגבל ל15 דק', בתום הזמן הזה יקבל המשתמש הודעה על סיום ההזמנה.

תודה רבה ובתאבון!

Form1:

```
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;

namespace McsOrder
{
    public enum FoodType
    {
        Sandwich,
        Salad,
        Dessert,
        Drink,
        Sausage
    }

    public partial class frmMcsOrder : Form
    {
        private DateTime startTime;

        // Create Cart instance
        Cart cart = new Cart();
        public frmMcsOrder()
        {
            InitializeComponent();

            cmbChooseTypeAddToCart.DataSource =
Enum.GetValues(typeof(FoodType)); // Load Enum FoodType to ComboBox
            cmbChooseTypeAddToCart.SelectedIndex = -1; // Clear selection
of the
        }

        private void timerShowTime_Tick(object sender, EventArgs e)
        {

        }
        //.ToString("hh:mm:ss")

        private void frmMcsOrder_Load(object sender, EventArgs e)
        {
            /* Init timer */
            timerShowTime.Interval = 1000; // Set the timer to work every
second (1000 miliSeconds)
            timerShowTime.Tick += timerShowTime_Tick; // Handle the
timer's Tick Event
            timerShowTime.Start(); // Start the timer
        }


       private void StartStopper()
        {
            timerStopper.Interval = 1000;
            timerStopper.Start();
```

```
            startTime = DateTime.Now;
            lblStopper.Text = "";
        }

        private void btnStartWelcome_Click(object sender, EventArgs e)
        {
            grpWelcome.Visible =true; // Hide the Welcome group
            btnStartWelcome.Visible = false;
            grpAddToCart.Visible = true; // Display Add to Cart group
            StartStopper();
            grpStopper.Visible = true;
            lblStopper.Visible = true;
        }

        private void cmbChooseTypeAddToCart_SelectedIndexChanged(object
sender, EventArgs e)
        {
            if (cmbChooseTypeAddToCart.SelectedItem != null)
            {
                // Get the Food Type selected in the ComboBox
                FoodType foodTypeSelected =
(FoodType)cmbChooseTypeAddToCart.SelectedItem;
                Food[] foodsSameType =
cart.GetFoodsFromSameType(foodTypeSelected);

                // Display foodsSameType in ListView
                lstFoodsSameTypeAddToCart.DataSource = foodsSameType;

                // Add to the Label title of Foods Same Type
                lblFoodsSameTypeAddToCart.Text = "Our " +
                    cmbChooseTypeAddToCart.SelectedItem.ToString() + "
Menu:";

                // Display ListView of the foods from the same type
                lblFoodsSameTypeAddToCart.Visible = true;
                lstFoodsSameTypeAddToCart.Visible = true;
            }
            else
            {
                lblFoodsSameTypeAddToCart.Text = "";
                lstFoodsSameTypeAddToCart.Visible = false; // Hide List
View of foods same type
            }
        }
```

```
private void lstFoodsSameTypeAddToCart_SelectedIndexChanged(object sender,
EventArgs e)
        {
            if (lstFoodsSameTypeAddToCart.SelectedItem != null) // Show
next elements
            {
                lblAmountAddToCart.Visible = true;
                txtAmontAddToCart.Clear();
                txtAmontAddToCart.Visible = true;
                btnCalculatePriceAddToCart.Enabled = false;
            }
            else // Hide next elements
            {
                lblAmountAddToCart.Visible = false;
                txtAmontAddToCart.Visible = false;
                lblPriceAddToCart.Visible = false;
                btnCalculatePriceAddToCart.Visible = false;
                btnSubmitAddToCart.Visible = false;
            }
        }

        private void txtAmontAddToCart_TextChanged(object sender,
EventArgs e)
        {
            // Enable and show Buttons Calculate and Submit
            btnCalculatePriceAddToCart.Enabled = true;
            btnCalculatePriceAddToCart.Visible = true;
            btnSubmitAddToCart.Enabled = true;
            btnSubmitAddToCart.Visible = true;
        }

        private void btnCalculatePriceAddToCart_Click(object sender,
EventArgs e)
        {
            // Input from TextBox AddToCart section
            string inputTxtBox = txtAmontAddToCart.Text;
            int quantityToCalc;

            // Try Parse in the input from TextBox
            bool isParsed = int.TryParse(inputTxtBox, out quantityToCalc);

            if (!isParsed) // Error in Parsing
            {
                MessageBox.Show("Wrong input, Error in parsing the input
into integer value.",
                    "error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                txtAmontAddToCart.Clear(); // Clear the TextBox of Amount
                btnSubmitAddToCart.Enabled = false; // Disable Submit
Button (Wrong input)
                lblPriceAddToCart.Visible = false;
            }
            else // Success in Parsing
            {
                // Check if positive
                if (quantityToCalc > 0)
                {
                    // Calculate the Price
                    Food selectedFood =
lstFoodsSameTypeAddToCart.SelectedItem as Food;
                    double result = selectedFood.GetUnitPrice() *
quantityToCalc;
                    lblPriceAddToCart.Text = $"Total Price is: {result}$";
                    lblPriceAddToCart.Visible = true;                }
```

```
                btnSubmitAddToCart.Enabled = true; // Enable Submit Button
(Wrong input)
            }

            btnCalculatePriceAddToCart.Enabled = false; // Disable the
button Calculate
        }

        private void btnSubmitAddToCart_Click(object sender, EventArgs e)
        {
            // Input from TextBox AddToCart section
            string inputTxtBox = txtAmontAddToCart.Text;
            int quantityAddToCart;

            // Try Parse in the input from TextBox
            bool isParsed = int.TryParse(inputTxtBox, out
quantityAddToCart);

            if (!isParsed) // Error in Parsing
            {
                MessageBox.Show("Wrong input, Enter positive integer
number.",
                    "error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                btnSubmitAddToCart.Enabled = false; // Disable Submit
Button (Wrong input)
            }
            else // Success in Parsing
            {
                // Get the Food from ListBox and create the OrderedFood
                Food foodSelected = lstFoodsSameTypeAddToCart.SelectedItem
as Food;
                // Add the OrderedFood to cart
                bool result = cart.AddFoodToCart(foodSelected,
quantityAddToCart);

                if (result) // success in Adding
                {

                    lstDisplayFoodsCart.Items.Add(new
OrderedFood(foodSelected, quantityAddToCart));
                    int numFoodsInCart = cart.GetNumOfItemsInCart(); //
Get the numInCart
                    double updatedPriceInCart =
cart.GetTotalPriceInCart();
                    if (numFoodsInCart == 1)
                        lblDataCalcCart.Text = $"There is 1 item in your
cart.\nTotal Price: {updatedPriceInCart}.";
                    else
                        lblDataCalcCart.Text = $"There are
{numFoodsInCart} items in your cart.\nTotal Price: {updatedPriceInCart}.";
                    lblDataCalcCart.Visible = true;

                    MessageBox.Show($"The {foodSelected.GetName()} was
added to your Cart!", "success");
                    grpCart.Visible = true; // Show GroupBox of Cart
                    lstDisplayFoodsCart.SelectedItem = null;
                }
```

```
                    else // Food is already in Cart
                    {
                        MessageBox.Show("Food exist in Cart, choose other
Food or remove it and add again.",
                        "error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                    }
            }

            if (!cart.IsCartEmpty()) // Cart is not empty
            {
                btnCheckoutCart.Visible = true;
                btnClearCart.Visible = true;
            }

            // Clear selected items in List Same Type
            lstFoodsSameTypeAddToCart.SelectedItem = null;
            txtAmontAddToCart.Clear(); // Clear the TextBox of Amount
        }
private void lstDisplayFoodsCart_SelectedIndexChanged(object sender,
EventArgs e)
        {
            if (lstDisplayFoodsCart.SelectedItem != null)
            {
                btnRemoveCart.Enabled = true;
                btnRemoveCart.Visible = true;
            }
            else
            {
                btnRemoveCart.Enabled = false;
                btnRemoveCart.Visible = false;
            }
        }

        private void btnRemoveCart_Click(object sender, EventArgs e)
        {
            // Get the selected OrderedFood
            OrderedFood orderedFoodToRemove =
lstDisplayFoodsCart.SelectedItem as OrderedFood;
            int indexSelected = (int)lstDisplayFoodsCart.SelectedIndex;

            if (indexSelected == -1)
            {
                MessageBox.Show("Please choose the item you want to
remove", "Error no selected item");
            }

            else
            {
                // Remove from Cart

cart.RemoveFoodFromCart(orderedFoodToRemove.GetFoodOrdered());
                lstDisplayFoodsCart.Items.RemoveAt(indexSelected);

                int numFoodsInCart = cart.GetNumOfItemsInCart(); // Get
the numInCart
                double updatedPriceInCart =
Math.Round(cart.GetTotalPriceInCart(), 12);
```

```
                if (numFoodsInCart == 1)
                    lblDataCalcCart.Text = $"There is 1 item in your
cart.\nTotal Price: {updatedPriceInCart}.";

                else
                    lblDataCalcCart.Text = $"There are {numFoodsInCart}
items in your cart.\nTotal Price: {updatedPriceInCart}.";

                MessageBox.Show($"The
{orderedFoodToRemove.GetFoodOrdered().GetName()} was removed from Cart.",
"success");

                if (cart.IsCartEmpty()) // If the last Food removed
                {
                    timerStopper.Stop();
                    grpCart.Visible = false;
                }
            }

            lstDisplayFoodsCart.SelectedItem = null;
            lblDataCalcCart.Visible = true;
        }

        private void btnCheckoutCart_Click(object sender, EventArgs e)
        {
            // Terminate Program and give the amount payed with MessageBox
            MessageBox.Show($"Thank for buying from us :)\nTotal payment:
" +
                $"{cart.GetTotalPriceInCart()}\nTime Order:
{lblStopper.Text}",
                "Good Bye", MessageBoxButtons.OK);
            timerStopper.Stop();
            this.Close();
        }

        private void btnClearCart_Click(object sender, EventArgs e)
        {
            // Clear the Cart
            cart.ClearCart();
            // Clear the list
            lstDisplayFoodsCart.Items.Clear();
            cmbChooseTypeAddToCart.SelectedItem = null;
            MessageBox.Show("Your cart was cleared.", "Clear Cart");
            timerStopper.Stop();
            grpAddToCart.Visible = false;
            grpCart.Visible = false;
            btnStartWelcome.Visible = true;
        }
```

```
        private void timer2_Tick(object sender, EventArgs e)
        {
            TimeSpan timeSpan = DateTime.Now-startTime;
            if (timeSpan.Seconds == 30)
            {
                timerStopper.Stop();

                MessageBox.Show($"You have reached to the maximum time of
Order\n" +
                    $"Which is: {timeSpan.Seconds} sec", "Order
finished");

                grpAddToCart.Visible = false;
                grpCart.Visible = false;
                btnStartWelcome.Visible = true;
                grpStopper.Visible = false;
                lstDisplayFoodsCart.Items.Clear();
                cart.ClearCart();
                cmbChooseTypeAddToCart.SelectedItem = null;
            }
            else
            {
                lblStopper.Text = string.Format("{0:mm\\:ss}",timeSpan);
            }
        }
    }
```

Cart:

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

namespace McsOrder
{
    internal class Cart
    {
        // Init static array of Foods in stock
        public static Food[] foodsInStock =
        {
            // Sandwich
            // Salad
            // Dessert
            // Drink
            // Sausage

            /* Sandwiches */
            new Food("Big Mc Gui", 10.25, 257.2, FoodType.Sandwich),
            new Food("Mc Gui with soy Cheese", 11.5, 244,
FoodType.Sandwich),
            new Food("Gui Cheeseburger", 12.29, 303.1, FoodType.Sandwich),
            new Food("McChicken", 10.99, 272.5, FoodType.Sandwich),
            new Food("Fish and Chips", 12.99, 282.5, FoodType.Sandwich),

            /* Salads */
            new Food("Beef Salad", 5.5, 495, FoodType.Salad),
            new Food("Dennis Salad", 5.05, 258, FoodType.Salad),
            new Food("Crispy Chicken Salad", 10.5, 318, FoodType.Salad),
            new Food("Chef Salad", 15.99, 184, FoodType.Salad),
            new Food("Israeli Salad", 8.69, 70, FoodType.Salad),
            new Food("Caesar Salad", 12.5, 184, FoodType.Salad),

            /* Desserts */
            new Food("Fruit Dessert", 12.5, 270, FoodType.Dessert),
            new Food("Gelatin Dessert", 10.99, 156, FoodType.Dessert),
            new Food("Baby Food Fruit Dessert", 15.5, 9,
FoodType.Dessert),
            new Food("Ice Cream", 20.99, 198, FoodType.Dessert),
            new Food("Ricotta Cheese Dessert", 11.5, 204,
FoodType.Dessert),
            new Food("Ladoo Round Ball", 13.99, 246, FoodType.Dessert),

            /* Drinks */
            new Food("Pepsi Soft", 8.5, 137, FoodType.Drink),
            new Food("Orange Juice", 6.5, 138, FoodType.Drink),
            new Food("Grape Juice", 9.2, 137, FoodType.Drink),
            new Food("7 Up ", 7.25, 137, FoodType.Drink),

            /* Sausages */
            new Food("Ketchup", 0.5, 111.6, FoodType.Sausage),
            new Food("Mayonnaise", 0.25, 679.7, FoodType.Sausage),
            new Food("Thousand Islands", 0.75, 114, FoodType.Sausage)

        };

        /* Props */
        static OrderedFood[] myCart;
```

```
        static double totalPriceInCart;

        /* Constructor */
        public Cart()
        {
            myCart = new OrderedFood[0];
            totalPriceInCart = 0;
        }

        public int GetNumOfItemsInCart() { return myCart.Length; }
        public bool IsCartEmpty() { return myCart.Length == 0; }

        /* Getters */
        public OrderedFood[] GetCart() { return myCart; }
        public double GetTotalPriceInCart() { return totalPriceInCart; }
        public void ClearCart()
        {
            Array.Clear(myCart, 0, myCart.Length);
            myCart = new OrderedFood[0];
            totalPriceInCart = 0;
        }
        public bool IsFoodExistInCart(Food foodToSearch)
        {
            if (foodToSearch == null)
                return false;

            foreach (OrderedFood orderedFood in myCart)
            {
                if (orderedFood != null)
                {
                    Food foodOrdered = orderedFood.GetFoodOrdered();
                    if (foodOrdered != null && foodOrdered.GetName() ==
foodToSearch.GetName())
                        return true;
                }
            }

            return false;
        }

        public int GetIndexOfFoodInCart(Food foodToSearch)
        {
            Food currentFood;
            for (int i = 0; i < myCart.Length; i++)
            {
                currentFood = myCart[i].GetFoodOrdered();
                if (currentFood.GetName() == foodToSearch.GetName())
                    return i;
            }
            return -1; // If didnt find the Food
        }

        /* Functions - operations */
        public bool AddFoodToCart(Food foodToAdd, int amountOrdered) //
Add to the end of the Cart
        {
            if (!IsFoodExistInCart(foodToAdd))
            {
                // Create new array
                OrderedFood[] tempArr = new OrderedFood[myCart.Length +
1];
                // Create new OrderFood
```

```
                OrderedFood orderedFoodToAdd = new OrderedFood(foodToAdd,
amountOrdered);

                // Copy myCart to tempArr
                Array.Copy(myCart, tempArr, myCart.Length);
                // Add Food at the end of Cart
                tempArr[myCart.Length] = orderedFoodToAdd;
                // Reassign myCart to tempArr
                myCart = tempArr;

                // Update totalPriceInCart
                totalPriceInCart += (foodToAdd.GetUnitPrice() *
amountOrdered);

                return true; // success if the Food does not exist in
myCart
            }

            return false; // failure if Food exists in myCart
        }

        public bool RemoveFoodFromCart(Food foodToRemove)
        {
            if (!IsCartEmpty())
            {
                // Get index of foodToRemove
                int indexOrderedFoodToRemove =
GetIndexOfFoodInCart(foodToRemove);

                if (indexOrderedFoodToRemove >= 0)
                {
                    // Create new array
                    OrderedFood[] tempArr = new OrderedFood[myCart.Length
- 1];

                    // Copy all the elements until index
                    for (int i = 0; i < indexOrderedFoodToRemove; i++)
                        tempArr[i] = myCart[i];
                    // Copy from the next index until the last
                    for (int i = indexOrderedFoodToRemove + 1; i <
myCart.Length; i++)
                        tempArr[i-1] = myCart[i];

                    int amountToRemove =
myCart[indexOrderedFoodToRemove].GetAmountOrdered();
                    double unitPriceToRemove =
myCart[indexOrderedFoodToRemove].GetFoodOrdered().GetUnitPrice();

                    // Update totalPriceInCart
                    totalPriceInCart -= (unitPriceToRemove *
amountToRemove);

                    // Reassign myCart to tempArr
                    myCart = tempArr;
                }

                return true;
            }

            return false; // If cart is empty
        }
```

```
        // Generate an array of foods from the same type (from
foodsInStock)
        public Food[] GetFoodsFromSameType(FoodType foodType)
        {
            List<Food> foodsSameType = new List<Food>();

            foreach (Food food in foodsInStock)
                if (food.GetFoodType() == foodType)
                    foodsSameType.Add(food);

            return foodsSameType.ToArray();
        }
    }
}
```

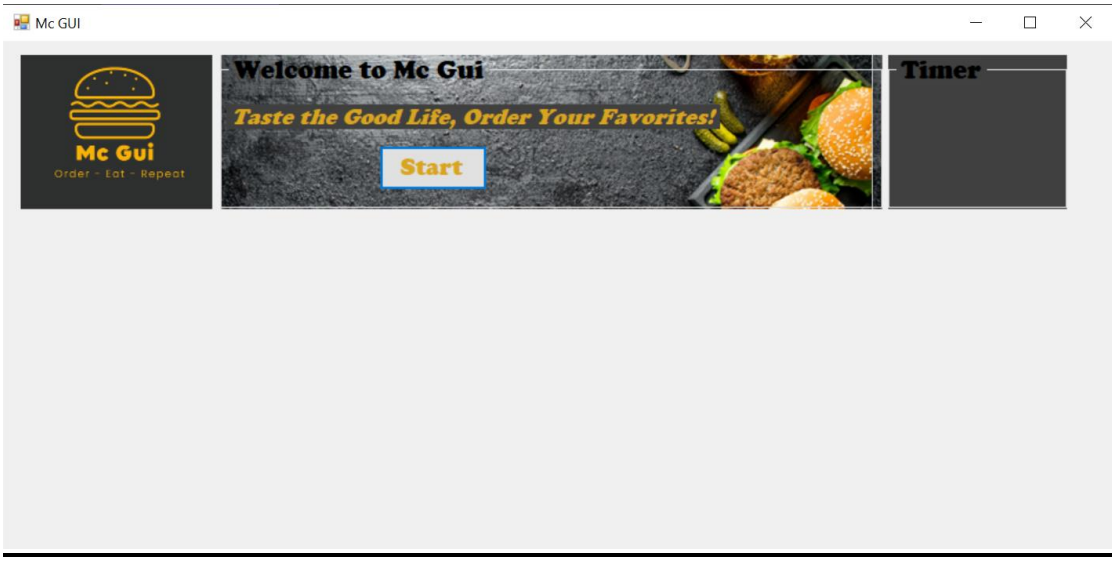
OrderedFood:

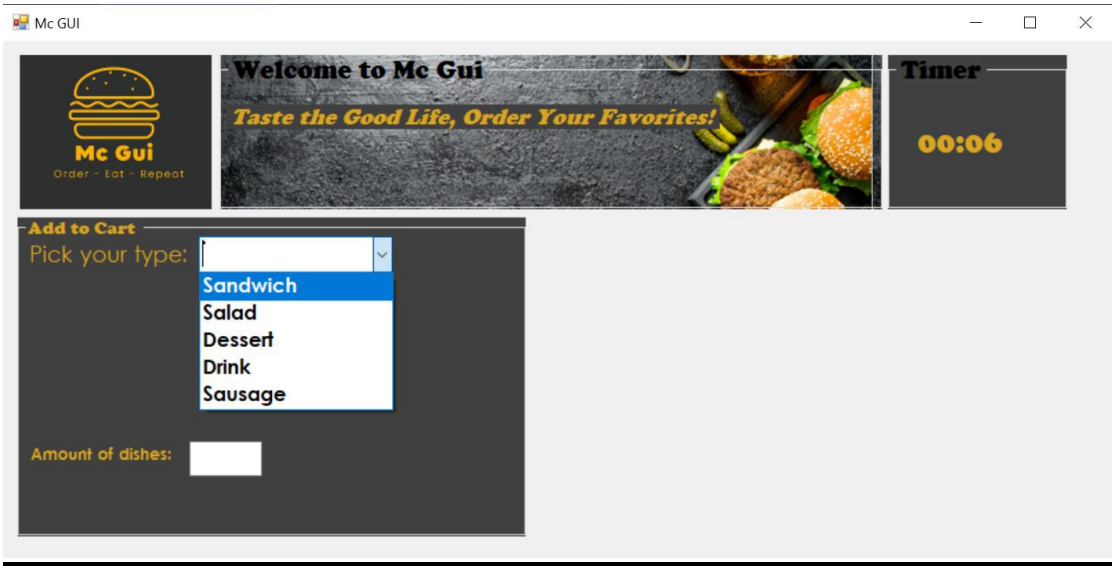
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace McsOrder
{
    internal class OrderedFood
    {
        /* Props */
        Food foodOrdered;
        int amountOrdered;

        /* Constructor */
        public OrderedFood(Food foodOrdered, int amountOrdered)
        {
            this.foodOrdered = foodOrdered;
            this.amountOrdered = amountOrdered;
        }

        /* Getters */
        public Food GetFoodOrdered() { return foodOrdered; }
        public int GetAmountOrdered() { return amountOrdered; }

        public void SetFoodOrdered(Food foodOrdered) { this.foodOrdered =
foodOrdered; }
        public void SetAmount(int amountOrdered) { this.amountOrdered=
amountOrdered; }

        /* ToString */
        public override string ToString()
        {
            return $"{foodOrdered.ToString()} , Amount: {amountOrdered}.";
        }
    }
}
```

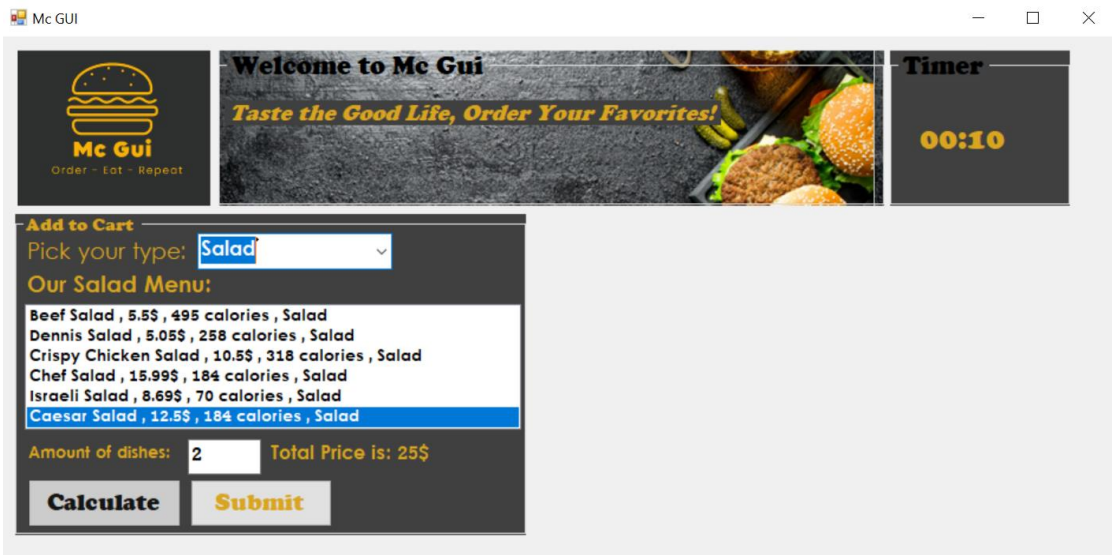
Food:

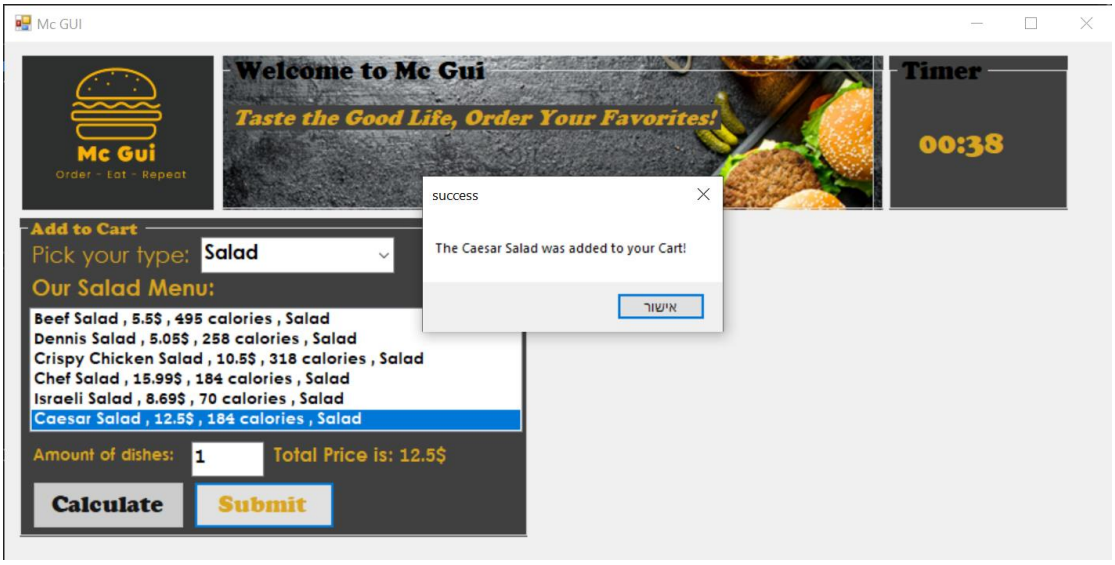
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace McsOrder
{
    internal class Food
    {
        /* Props */
        string name;
        double unitPrice;
        double calories;
        FoodType foodType;

        /* Constructor */
        public Food(string name, double unitPrice, double calories,
FoodType foodType)
        {
            this.name = name;
            this.unitPrice = unitPrice;
            this.calories = calories;
            this.foodType = foodType;
        }

        /* Getters */
        public string GetName() { return name; }
        public double GetUnitPrice() { return unitPrice; }
        public double GetCalories() { return calories; }
        public FoodType GetFoodType() { return foodType; }

        /* ToString Method */
        public override string ToString()
        {
            return $"{name} , {unitPrice}$ , {calories} calories ,
{foodType}";
        }
    }
}
```

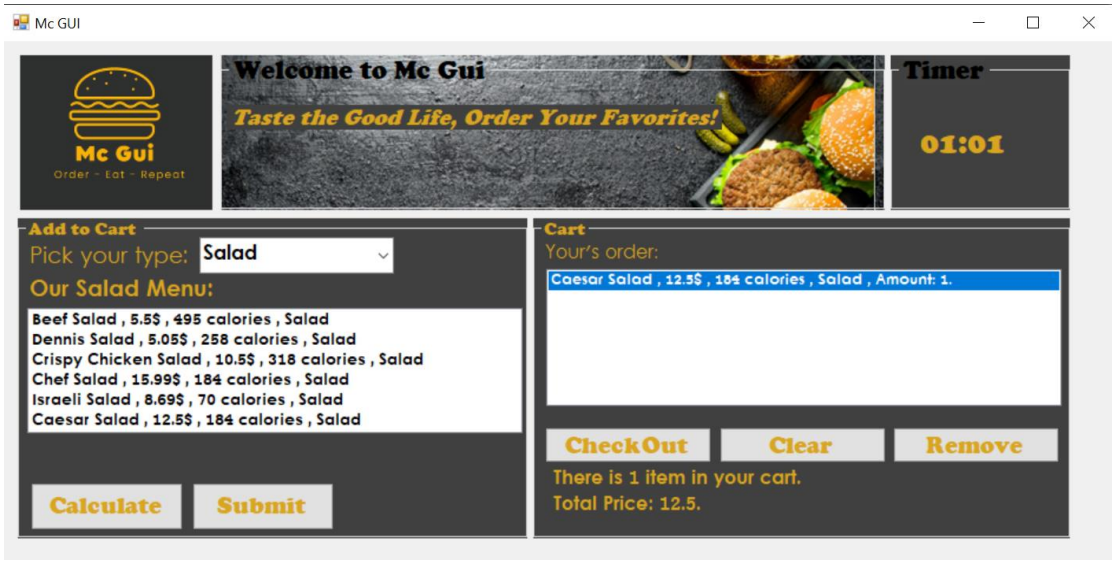
Outputs Q2:

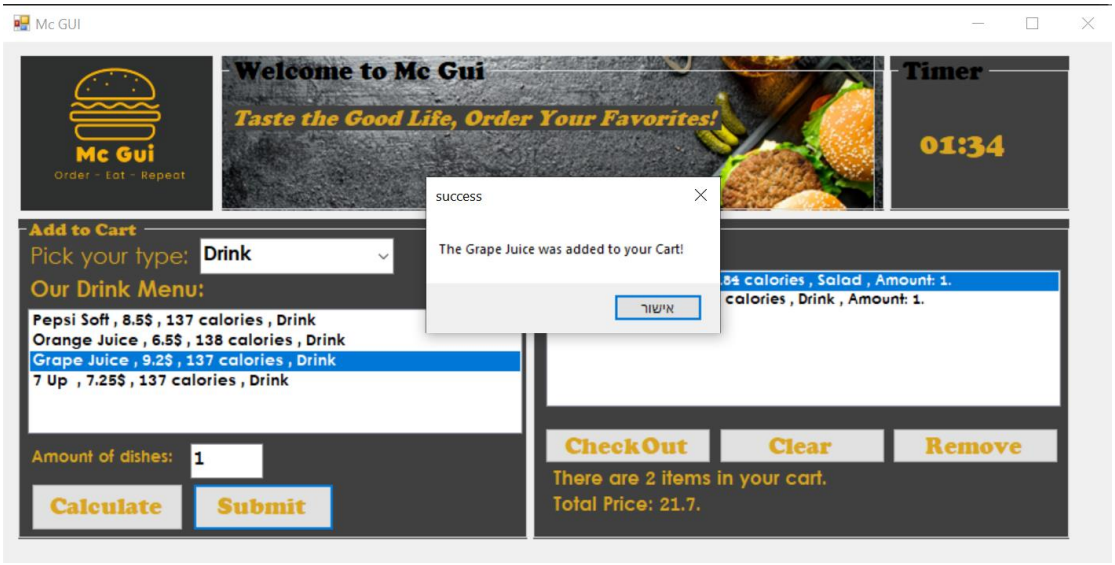

Mc GUI
Mc Gui
Order - Eat - Repeat
Welcome to Mc Gui
Taste the Good Life, Order Your Favorites!
Start
Timer

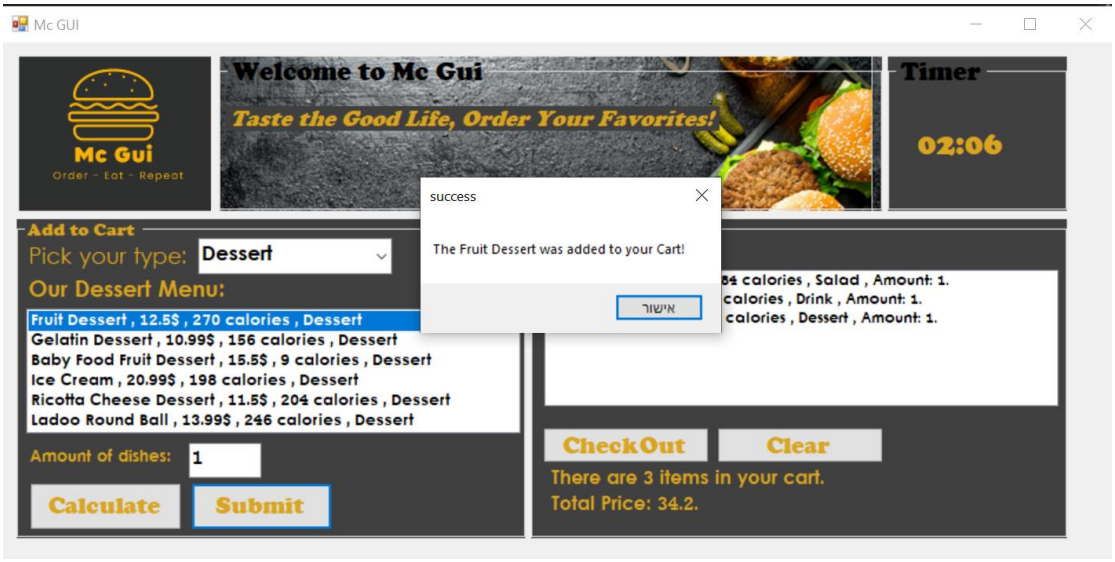

Mc GUI
Mc Gui
Order - Eat - Repeat
Welcome to Mc Gui
Taste the Good Life, Order Your Favorites!
Timer
00:06
Add to Cart
Pick your type:
Sandwich
Salad
Dessert
Drink
Sausage
Amount of dishes:

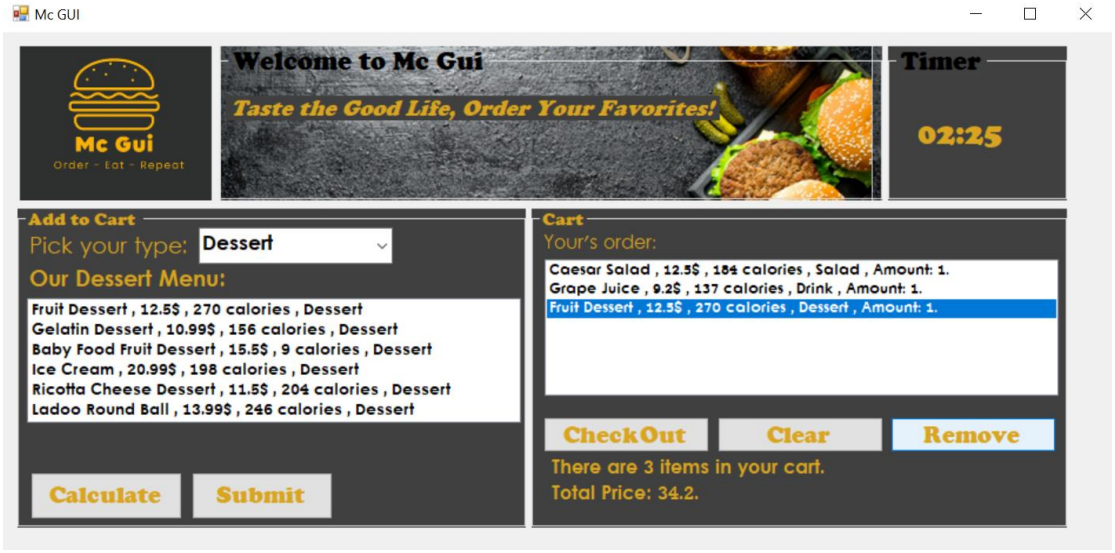

Mc GUI
Mc Gui
Order - Eat - Repeat
Welcome to Mc Gui
Taste the Good Life, Order Your Favorites!
Timer
00:10
Add to Cart
Pick your type:
Salad
Our Salad Menu:
Beef Salad , 5.5$, 495 calories , Salad
Dennis Salad , 5.05$, 258 calories , Salad
Crispy Chicken Salad , 10.5$, 318 calories , Salad
Chef Salad , 15.99$, 184 calories , Salad
Israeli Salad , 8.69$, 70 calories , Salad
Caesar Salad , 12.5$, 184 calories , Salad
Amount of dishes:
2
Total Price is: 25$
Calculate
Submit

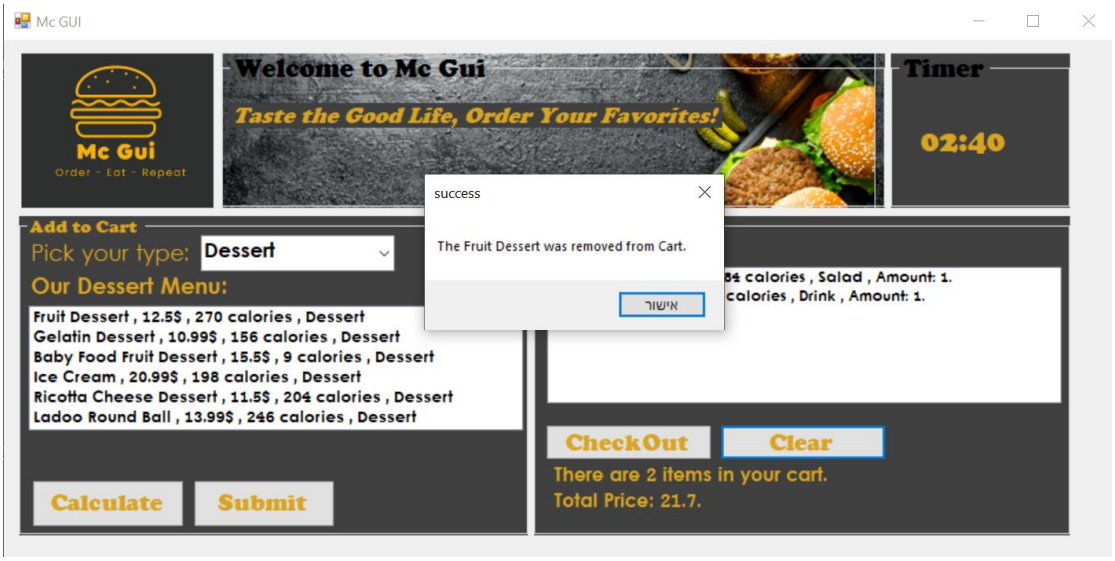

Mc GUI
Mc Gui
Order - Eat - Repeat
Welcome to Mc Gui
Taste the Good Life, Order Your Favorites!
Timer
00:38
success
The Caesar Salad was added to your Cart!
אישור
Add to Cart
Pick your type: Salad
Our Salad Menu:
Beef Salad , 5.5$, 495 calories , Salad
Dennis Salad , 5.05$, 258 calories , Salad
Crispy Chicken Salad , 10.5$, 318 calories , Salad
Chef Salad , 15.99$, 184 calories , Salad
Israeli Salad , 8.69$, 70 calories , Salad
Caesar Salad , 12.5$, 184 calories , Salad
Amount of dishes: 1
Total Price is: 12.5$
Calculate
Submit

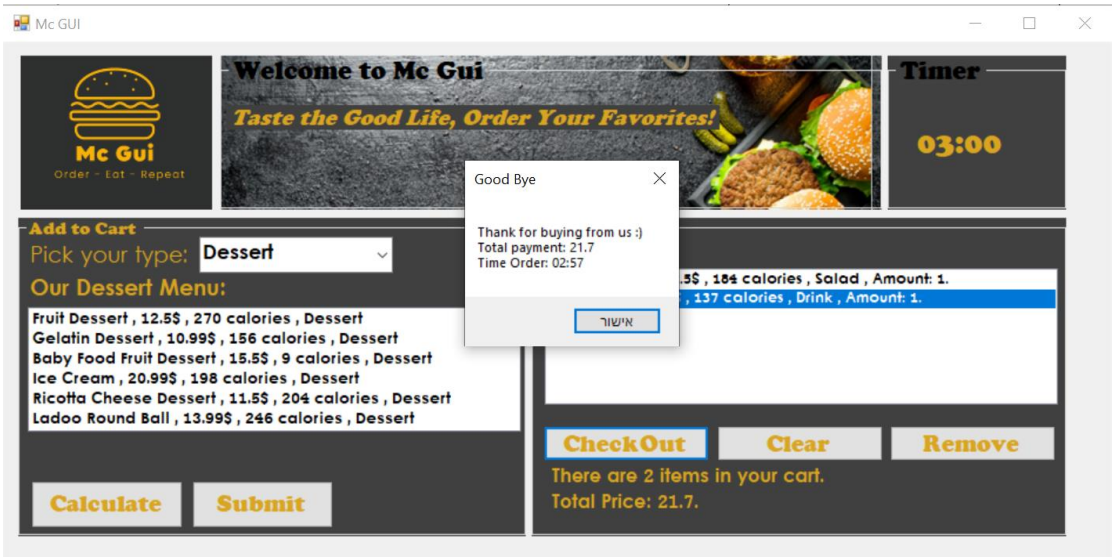

Mc GUI
Mc Gui
Order - Eat - Repeat
Welcome to Mc Gui
Taste the Good Life, Order Your Favorites!
Timer
01:01
Add to Cart
Pick your type: Salad
Our Salad Menu:
Beef Salad , 5.5$, 495 calories , Salad
Dennis Salad , 5.05$, 258 calories , Salad
Crispy Chicken Salad , 10.5$, 318 calories , Salad
Chef Salad , 15.99$, 184 calories , Salad
Israeli Salad , 8.69$, 70 calories , Salad
Caesar Salad , 12.5$, 184 calories , Salad
Calculate
Submit
Cart
Your's order:
Caesar Salad , 12.5$, 184 calories , Salad , Amount: 1.
CheckOut
Clear
Remove
There is 1 item in your cart.
Total Price: 12.5.

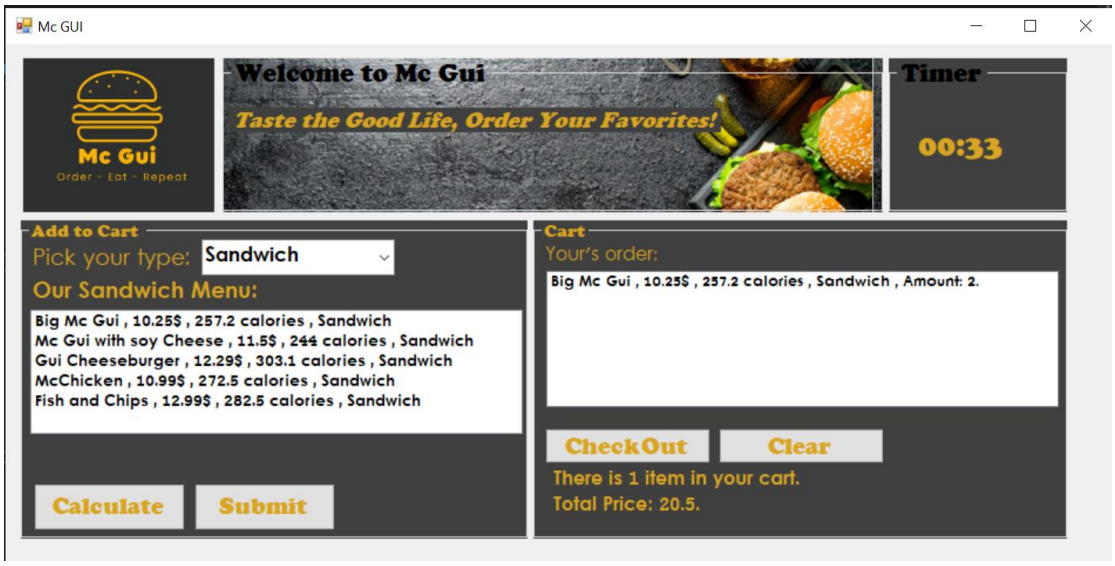

Mc GUI
Mc Gui
Order - Eat - Repeat
Welcome to Mc Gui
Taste the Good Life, Order Your Favorites!
Timer
01:34
success
The Grape Juice was added to your Cart!
אישור
Add to Cart
Pick your type: Drink
Our Drink Menu:
Pepsi Soft , 8.5$, 137 calories , Drink
Orange Juice , 6.5$, 138 calories , Drink
Grape Juice , 9.2$, 137 calories , Drink
7 Up , 7.25$, 137 calories , Drink
Amount of dishes: 1
Calculate
Submit
84 calories , Salad , Amount: 1.
calories , Drink , Amount: 1.
CheckOut
Clear
Remove
There are 2 items in your cart.
Total Price: 21.7.

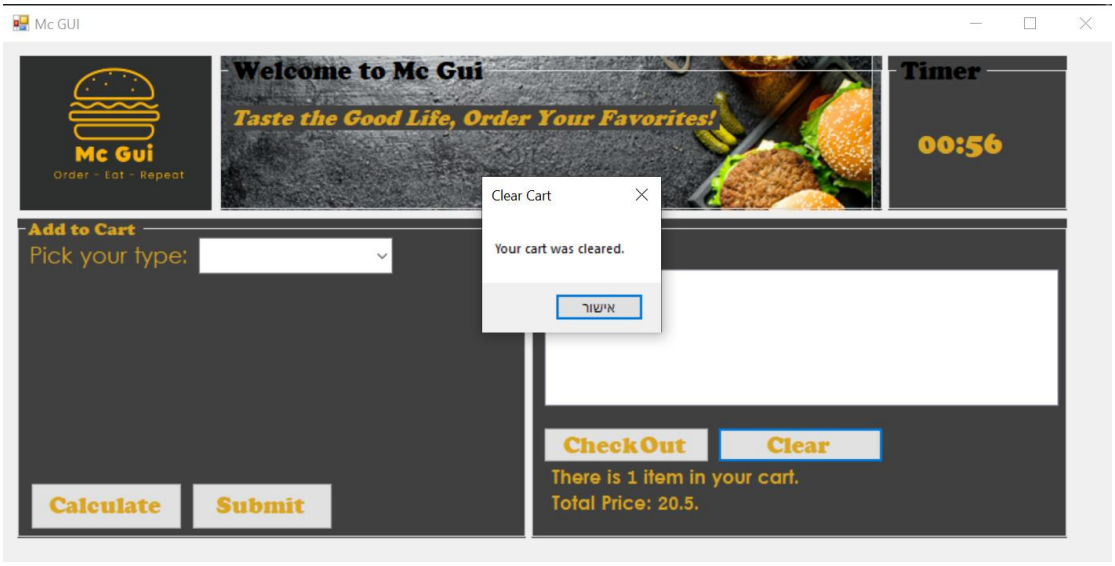

Mc GUI
Mc Gui
Order - Eat - Repeat
Welcome to Mc Gui
Taste the Good Life, Order Your Favorites!
Timer
02:06
success
The Fruit Dessert was added to your Cart!
אישור
Add to Cart
Pick your type:
Dessert
Our Dessert Menu:
Fruit Dessert , 12.5$, 270 calories , Dessert
Gelatin Dessert , 10.99$, 156 calories , Dessert
Baby Food Fruit Dessert , 15.5$, 9 calories , Dessert
Ice Cream , 20.99$, 198 calories , Dessert
Ricotta Cheese Dessert , 11.5$, 204 calories , Dessert
Ladoo Round Ball , 13.99$, 246 calories , Dessert
Amount of dishes:
1
Calculate
Submit
84 calories , Salad , Amount: 1.
calories , Drink , Amount: 1.
calories , Dessert , Amount: 1.
CheckOut
Clear
There are 3 items in your cart.
Total Price: 34.2.

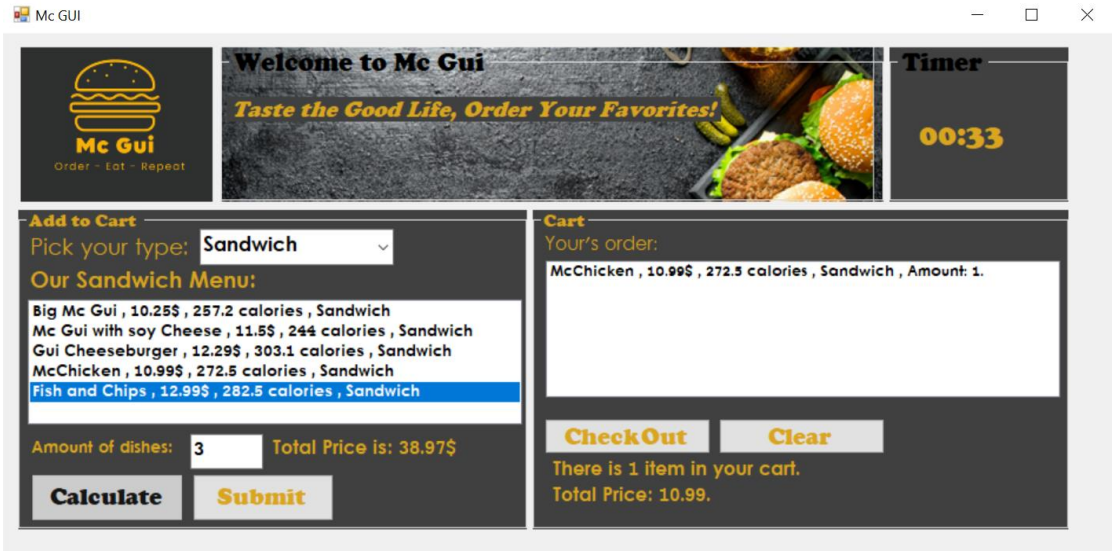

Mc GUI
Mc Gui
Order - Eat - Repeat
Welcome to Mc Gui
Taste the Good Life, Order Your Favorites!
Timer
02:25
Add to Cart
Pick your type:
Dessert
Our Dessert Menu:
Fruit Dessert , 12.5$, 270 calories , Dessert
Gelatin Dessert , 10.99$, 156 calories , Dessert
Baby Food Fruit Dessert , 15.5$, 9 calories , Dessert
Ice Cream , 20.99$, 198 calories , Dessert
Ricotta Cheese Dessert , 11.5$, 204 calories , Dessert
Ladoo Round Ball , 13.99$, 246 calories , Dessert
Calculate
Submit
Cart
Your's order:
Caesar Salad , 12.5$, 184 calories , Salad , Amount: 1.
Grape Juice , 9.2$, 137 calories , Drink , Amount: 1.
Fruit Dessert , 12.5$, 270 calories , Dessert , Amount: 1.
CheckOut
Clear
Remove
There are 3 items in your cart.
Total Price: 34.2.

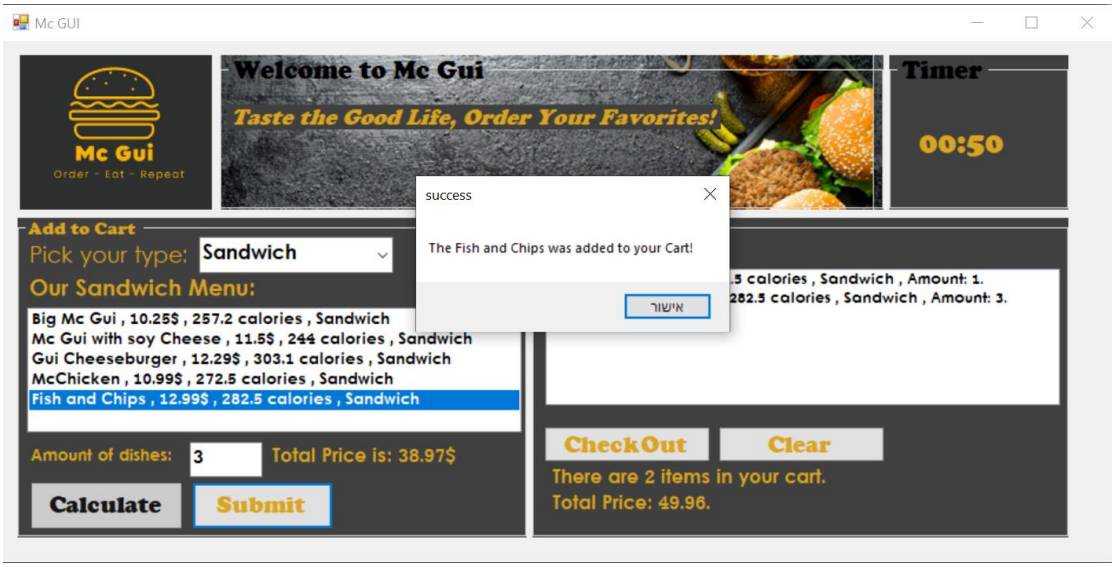

Mc GUI
Mc Gui
Order - Eat - Repeat
Welcome to Mc Gui
Taste the Good Life, Order Your Favorites!
Timer
02:40
success
The Fruit Dessert was removed from Cart.
אישור
Add to Cart
Pick your type:
Dessert
Our Dessert Menu:
Fruit Dessert , 12.5$, 270 calories , Dessert
Gelatin Dessert , 10.99$, 156 calories , Dessert
Baby Food Fruit Dessert , 15.5$, 9 calories , Dessert
Ice Cream , 20.99$, 198 calories , Dessert
Ricotta Cheese Dessert , 11.5$, 204 calories , Dessert
Ladoo Round Ball , 13.99$, 246 calories , Dessert
Calculate
Submit
84 calories , Salad , Amount: 1.
calories , Drink , Amount: 1.
CheckOut
Clear
There are 2 items in your cart.
Total Price: 21.7.

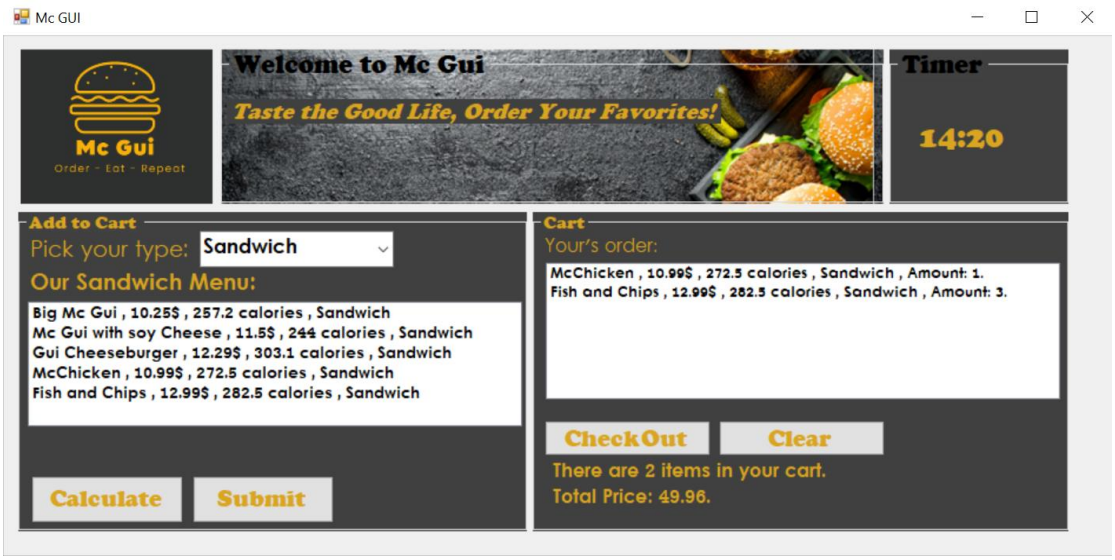

Mc GUI
Mc Gui
Order - Eat - Repeat
Welcome to Mc Gui
Taste the Good Life, Order Your Favorites!
Timer
03:00
Good Bye
Thank for buying from us :)
Total payment: 21.7
Time Order: 02:57
אישור
Add to Cart
Pick your type:
Dessert
Our Dessert Menu:
Fruit Dessert , 12.5$, 270 calories , Dessert
Gelatin Dessert , 10.99$, 156 calories , Dessert
Baby Food Fruit Dessert , 15.5$, 9 calories , Dessert
Ice Cream , 20.99$, 198 calories , Dessert
Ricotta Cheese Dessert , 11.5$, 204 calories , Dessert
Ladoo Round Ball , 13.99$, 246 calories , Dessert
Calculate
Submit
5$, 184 calories , Salad , Amount: 1.
, 137 calories , Drink , Amount: 1.
CheckOut
Clear
Remove
There are 2 items in your cart.
Total Price: 21.7.

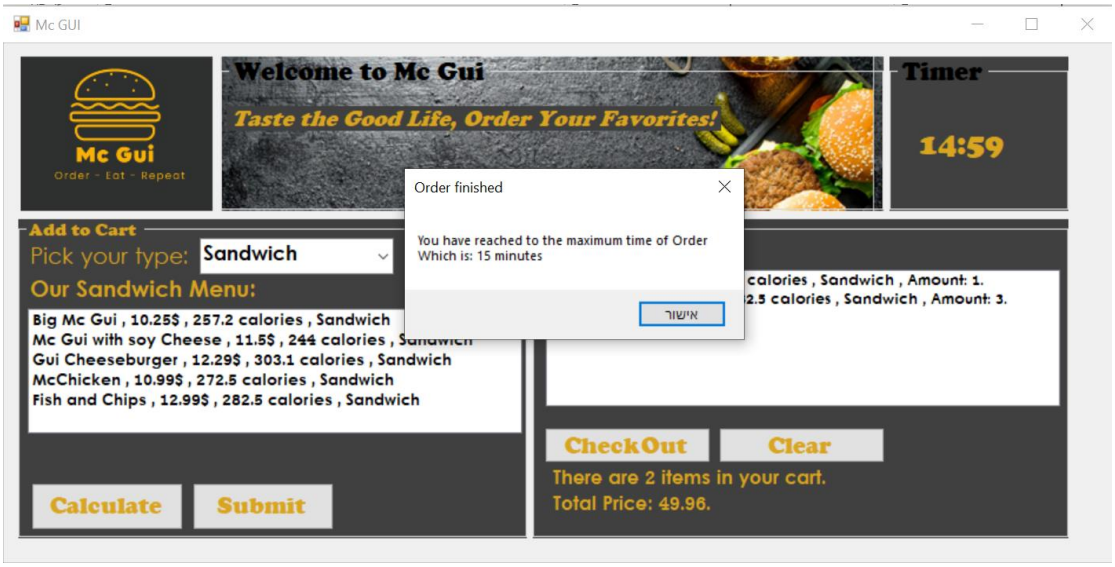

Mc GUI
Mc Gui
Order - Eat - Repeat
Welcome to Mc Gui
Taste the Good Life, Order Your Favorites!
Timer
00:33
Add to Cart
Pick your type:
Sandwich
Our Sandwich Menu:
Big Mc Gui , 10.25$, 257.2 calories , Sandwich
Mc Gui with soy Cheese , 11.5$, 244 calories , Sandwich
Gui Cheeseburger , 12.29$, 303.1 calories , Sandwich
McChicken , 10.99$, 272.5 calories , Sandwich
Fish and Chips , 12.99$, 282.5 calories , Sandwich
Calculate
Submit
Cart
Your's order:
Big Mc Gui , 10.25$, 257.2 calories , Sandwich , Amount: 2.
CheckOut
Clear
There is 1 item in your cart.
Total Price: 20.5.


Mc GUI
Mc Gui
Order - Eat - Repeat
Welcome to Mc Gui
Taste the Good Life, Order Your Favorites!
Timer
00:56
Clear Cart
Your cart was cleared.
אישור
Add to Cart
Pick your type:
Calculate
Submit
CheckOut
Clear
There is 1 item in your cart.
Total Price: 20.5.


Mc GUI
Mc Gui
Order - Eat - Repeat
Welcome to Mc Gui
Taste the Good Life, Order Your Favorites!
Timer
00:33
Add to Cart
Pick your type:
Sandwich
Our Sandwich Menu:
Big Mc Gui , 10.25$, 257.2 calories , Sandwich
Mc Gui with soy Cheese , 11.5$, 244 calories , Sandwich
Gui Cheeseburger , 12.29$, 303.1 calories , Sandwich
McChicken , 10.99$, 272.5 calories , Sandwich
Fish and Chips , 12.99$, 282.5 calories , Sandwich
Amount of dishes:
3
Total Price is: 38.97$
Calculate
Submit
Cart
Your's order:
McChicken , 10.99$, 272.5 calories , Sandwich , Amount: 1.
CheckOut
Clear
There is 1 item in your cart.
Total Price: 10.99.


Mc GUI
Mc Gui
Order - Eat - Repeat
Welcome to Mc Gui
Taste the Good Life, Order Your Favorites!
Timer
00:50
success
The Fish and Chips was added to your Cart!
אישור
Add to Cart
Pick your type:
Sandwich
Our Sandwich Menu:
Big Mc Gui , 10.25$, 257.2 calories , Sandwich
Mc Gui with soy Cheese , 11.5$, 244 calories , Sandwich
Gui Cheeseburger , 12.29$, 303.1 calories , Sandwich
McChicken , 10.99$, 272.5 calories , Sandwich
Fish and Chips , 12.99$, 282.5 calories , Sandwich
Amount of dishes:
3
Total Price is: 38.97$
Calculate
Submit
.5 calories , Sandwich , Amount: 1.
282.5 calories , Sandwich , Amount: 3.
CheckOut
Clear
There are 2 items in your cart.
Total Price: 49.96.


Mc GUI
Mc Gui
Order - Eat - Repeat
Welcome to Mc Gui
Taste the Good Life, Order Your Favorites!
Timer
14:20
Add to Cart
Pick your type:
Sandwich
Our Sandwich Menu:
Big Mc Gui , 10.25$, 257.2 calories , Sandwich
Mc Gui with soy Cheese , 11.5$, 244 calories , Sandwich
Gui Cheeseburger , 12.29$, 303.1 calories , Sandwich
McChicken , 10.99$, 272.5 calories , Sandwich
Fish and Chips , 12.99$, 282.5 calories , Sandwich
Calculate
Submit
Cart
Your's order:
McChicken , 10.99$, 272.5 calories , Sandwich , Amount: 1.
Fish and Chips , 12.99$, 282.5 calories , Sandwich , Amount: 3.
CheckOut
Clear
There are 2 items in your cart.
Total Price: 49.96.


Mc GUI
Mc Gui
Order - Eat - Repeat
Welcome to Mc Gui
Taste the Good Life, Order Your Favorites!
Timer
14:59
Order finished
You have reached to the maximum time of Order
Which is: 15 minutes
אישור
Add to Cart
Pick your type:
Sandwich
Our Sandwich Menu:
Big Mc Gui , 10.25$, 257.2 calories , Sandwich
Mc Gui with soy Cheese , 11.5$, 244 calories ,
Gui Cheeseburger , 12.29$, 303.1 calories , Sandwich
McChicken , 10.99$, 272.5 calories , Sandwich
Fish and Chips , 12.99$, 282.5 calories , Sandwich
calories , Sandwich , Amount: 1.
2.5 calories , Sandwich , Amount: 3.
Calculate
Submit
CheckOut
Clear
There are 2 items in your cart.
Total Price: 49.96.